WorkToDo Flex - Um Sistema de Gerenciamento de Workflows Flexíveis

Jackson da Cruz

Dissertação de Mestrado

Instituto de Computação
Universidade Estadual de Campinas

WorkToDo Flex - Um Sistema de Gerenciamento de Workflows Flexíveis

Jackson da Cruz

Fevereiro de 2005

Banca Examinadora:

- Prof. Dra. Maria Beatriz Felgar de Toledo
  Instituto de Computação, Unicamp (Orientadora)
- Prof. Dr. Edmundo Roberto Mauro Madeira
  Instituto de Computação, Unicamp
- Profa. Dra. Elisa Hatsue Moriya Huzita
  Departamento de Informática, UEM
- Profa. Dra. Anamaria Gomide
  Instituto de Computação, Unicamp (Suplente)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Cruz, Jackson da

C889w Worktodo flex – Um sistema de gerenciamento de workflows flexíveis / Jackson da Cruz -- Campinas, [S.P. :s.n.], 2005.

Orientador : Maria Beatriz Felgar de Toledo

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Fluxo de trabalho. 2. Software – Desenvolvimento. 3. Grupos de trabalho. I. Toledo, Maria Beatriz Felgar de. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Título em inglês: Worktodo flex – A Flexible workflow management system.

Palavras-chave em inglês (keywords): 1. Workflow. 2. Software – Development. 3. Workgroup.

Área de concentração: Sistemas distribuídos

Titulação: Mestre em Ciência da Computação

Banca examinadora: Profa. Dra. Maria Beatriz Felgar de Toledo (IC/UNICAMP)
Prof. Dr. Edmundo Roberto Mauro Madeira (IC/UNICAMP)
Profa. Dra. Elisa Hatsue Moriya Huzita (UEM)

Data da defesa: 21/02/2005

WorkToDo Flex - Um Sistema de Gerenciamento de Workflows Flexíveis

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Jackson da Cruz e aprovada pela Banca Examinadora.

Campinas, Fevereiro de 2005.

Prof. Dra. Maria Beatriz Felgar de Toledo
Instituto de Computação, Unicamp (Orientadora)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Agradecimentos

Primeiramente a Deus, por não deixar faltar força e vontade para que este trabalho fosse concluído.

Aos meus pais Antônio e Maria por acreditarem sempre na minha força de vontade e pelo apoio incondicional em todos os momentos da minha vida.

Aos meus irmãos Jeff, Katy e Deko pelo apoio, nos momentos de necessidade, e pelos conselhos nos momentos em que precisei.

Aos meus amigos, Evandro, Giovani, Juliana, por me aguentarem durante 1 ano morando na mesma casa e pela ajuda nos momentos de necessidade. Aos amigos Fernando, Alexandre (Capitão), Gregório, Borin, pelo companheirismo e a todos os demais amigos que de alguma forma sempre estiveram ao meu lado nos momentos de festa, churrascos, bebedeiras, e também nos momentos de responsabilidade.

À professora orientadora, Dra. Maria Beatriz Felgar de Toledo, pela paciência pelo fato da orientação ter se restringido a vindas periódias a Campinas, por aceitar ser minha orientadora, pelas sugestões sempre construtivas e, ao contrário de muitos orientadores, ser humana, humilde e respeitadora.

Aos colegas e amigos da UNIPAR e UNIOESTE pela compreensão e ajuda nos momentos em que tive que me ausentar para realizar as viajens à Campinas.

A todos aqueles que não impediram a conclusão desse trabalho.

Resumo

Processos de negócio de uma organização, normalmente, sofrem mudanças freqüentes resultantes de aspectos como a consolidação de seus sistemas e procedimentos, o melhoramento do processo de negócio, a necessidade de obedecer novas leis ou regimentos e reestruturação. Para acomodar tais mudanças, os módulos de especificação e execução de um sistema de gerenciamento de workflow devem estar estreitamente ligados. Por esses motivos, novos mecanismos de controle se fazem necessários, a fim de que o SGWf comporte todas as mudanças que o processo organizacional de uma empresa exija, sem comprometer a execução correta do processo. Estes mecanismos devem ser simples de forma a não causar grandes custos à empresa e permitir a rápida readequação de processos. Nesta dissertação apresentamos o WorkToDo Flex, um SGWf para workflows flexíveis que provê estes mecanismos de mudança através de dois tipos de flexibilidade: por adaptação e por seleção. A flexibilidade por adaptação é fornecida por um conjunto de operações realizadas em instâncias de processo em execução. A flexibilidade por seleção é obtida pela especificação e execução de tarefas de construção.

Abstract

Business processes of an organization may suffer many changes due to the consolidation of its systems and procedures, the improvement of the business process, the necessity to obey to new laws or regiments and reorganization. To accomodate such changes, the specification and execution modules of the WfMS must be linked. For this reason, new mechanisms of control are necessary, so that the WfMS supports all the changes that a business process of an organization demands, without compromising the correct execution of the process. These mechanisms must be simple so that cause great costs are avoided and fast reorganization of processes are allowed. In this dissertation we present the WorkToDo Flex, a WfMS for flexible workflows that provides two types of flexibility: adaptation and selection. Flexibility by adaptation is supplied by a set of change operations applied in process instances. Flexibility by selection is supplied by the specification and execution of task builders.

Nunca desista. Coloque no Easy.
(Gregorio Bagio Tramontina)

Sumário

Lista de Tabelas

Lista de Figuras

Tabela de Abreviaturas e Siglas

BNF	Backus Normal Form
CORBA	Common Object Request Broker Architecture
DM	Definition Manager
IDL	Interface Definition Language
IIOP	Internet Inter-ORB Protocol
IM	Instance Manager
JDK	Java Developer Kit
LDP	Linguagem de Definição de Processo
ORB	Object Request Broker
PM	Process Manager
RMI	Remote Method Invocation
SGWf	Sistema de Gerenciamento de Workflows
SPM	Sub-Process Manager
SQL	Simple Query Language
TM	Task Manager
UML	Unified Modeling Language
URM	User and Role Manager
WfMC	Workflow Management Coalition
WM	Worklist Manager

Capítulo 1

Introdução

1.1 Motivação

Constantemente as empresas buscam reduzir custos e aumentar lucros. Longe de ser apenas um sonho capitalista, esse panorama de crescimento tem se tornado necessário para que tais empresas mantenham-se ativas num mercado que, a cada dia, torna-se mais competitivo e globalizado. É questão de sobrevivência atualizar-se e se manter adiante dos concorrentes.

Surge então, a necessidade de uma análise mais profunda dos processos organizacionais que compõem essas empresas. Isto é, era preciso mais do que simples sistemas de automação comercial para controlar o fluxo de caixa e o estoque dos produtos.

Logo, para suprir estas necessidades, nascem dois conceitos: a reengenharia, que consiste em uma análise aprofundada dos processos organizacionais para repensá-los, melhorá-los e reimplantá-los, e workflow. Este último está relacionado com a automação dos processos propriamente dita, e evoluiu tanto que hoje podemos nos referir a ele como um conjunto de tecnologias e metodologias de automação de processos: a tecnologia de workflow.

Uma das principais motivações para se pesquisar essa tecnologia vem do fato de que ela permite, quando aplicada de maneira correta, atingir os objetivos anteriormente mencionados, trazendo assim benefícios financeiros para quem a fomenta. Ferramentas chamadas de Sistemas de Gerenciamento de Workflows (SGWfs) gerenciam a execução de processos empresariais, assumindo as funções mecânicas e burocráticas e permitindo que os usuários se concentrem em aspectos mais importantes, como gerenciamento de informações e análise de resultados [26].

Muitos SGWfs são vendidos comercialmente e passaram a ser utilizados nas empresas, alcançando significativas melhoras de eficiência. Entretanto, os SGWfs existentes ainda apresentam limitações como, por exemplo, baixa interoperabilidade entre diferentes SGWfs, apoio insuficiente para recuperação de falhas e pouca flexibilidade de uso e adaptação [3, 8, 10, 25].

Normalmente, as empresas sofrem mudanças freqüentes resultantes de aspectos como a consolidação de seus sistemas e procedimentos, o melhoramento do processo de negócio, a

necessidade de obedecer novas leis ou regimentos e reestruturação. Para acomodar tais mudanças, os módulos de especificação e execução de um sistema de gerenciamento de workflow devem estar estreitamente ligados. Por exemplo, deve ser possível editar a especificação de um procedimento no workflow e então, dinamicamente e com segurança, realizar a mudança nas instâncias em execução [9].

Podemos definir tal sistema de gerenciamento de workflow como "Flexível", ou seja, que permite modificações na especificação de processos de workflow e dinamicamente as acomoda nas instâncias em execução.

Por esses motivos, novos mecanismos de controle se fazem necessários, a fim de que o SGWf comporte todas as mudanças que o processo organizacional de uma empresa exija, sem comprometer a execução correta do processo. Estes mecanismos devem ser simples de forma a não causar grandes custos à empresa e permitir a rápida readequação de processos.

1.2 Objetivos

Nesta dissertação apresentamos o WorkToDo Flex, um SGWf distribuído que permite que usuários dispersos possam utilizar o sistema de maneira simples através de uma interface gráfica.

O principal objetivo é fornecer um SGWf que, além de gerenciar o fluxo de execução dos processos organizacionais de uma empresa, ofereça funcionalidades para permitir as mudanças dinâmicas que estes processos exijam.

O WorkToDo Flex implementa as funcionalidades que oferecem flexibilidade por seleção e adaptação [11]. O protótipo desenvolvido utiliza Java e RMI para avaliar o modelo e as funcionalidades de flexibilidade.

1.3 Estrutura da Dissertação

Este documento está organizado da seguinte forma:

- O capítulo 2 apresenta os conceitos necessários para o entendimento da dissertação, relacionados a Sistemas de Gerenciamento de Workflow e Flexibilidade;
- O capítulo 3 descreve o modelo e a arquitetura do WorkToDo, apresentando suas características e seu funcionamento;
- No capítulo 4 são descritos aspectos relativos às operações de flexibilidade implementadas no protótipo;
- Finalmente, o capítulo 5 apresenta as conclusões do trabalho, juntamente com uma proposta de alguns trabalhos futuros.

Capítulo 2

Sistemas de Gerenciamento de Workflow

Este capítulo descreve Sistemas de Gerenciamento de Workflow (SGWf). Na seção 2.1, são apresentados conceitos básicos e componentes da arquitetura proposta pela WfMC. A seção seguinte trata do problema de flexibilidade em SGWfs.

2.1 Conceitos Básicos

O termo workflow está relacionado com a automação de procedimentos organizacionais, cujos documentos, informações e tarefas são passadas entre as entidades participantes do processo, de acordo com um conjunto de regras, de forma que se atinja o objetivo desejado pela organização [26]. Estes procedimentos podem ter um ciclo de vida que varia de minutos até dias (ou eventualmente meses), dependendo da complexidade e duração de suas tarefas constituintes.

Um workflow pode ser executado utilizando-se tanto de técnicas “manuais”, isto é, sem a intervenção de computadores ou ferramentas de software, ou com a utilização dessas últimas. E, é claro, a última forma é a mais difundida. Podemos dizer até que nem mesmo podemos separar claramente o conceito de workflow da sua implementação e execução por ferramentas computadorizadas, devido à difusão dessas ferramentas e aos benefícios de sua utilização.

Um Sistema de Gerenciamento de Workflows (SGWf) é um sistema que define, gerencia e executa workflows de acordo com suas representações computacionais [26].

Podemos dizer que os SGWfs são inerentemente distribuídos, já que as entidades participantes podem estar espalhadas dentro de um edifício, uma cidade, um país, ou até mesmo ao redor do mundo.

O aspecto distribuição acima se refere à localização dos diversos componentes de software de um SGWf que podem executar em máquinas distintas e comunicar-se através de mensagens. Com relação ao controle de cada processo de workflow existem sistemas em que o controle está localizado em uma determinada máquina (por exemplo, o sistema WorkToDo [21][22]) e outros em que o controle é realizado de forma distribuída (por exemplo, INCAS [4][5]).

Mas mesmo não havendo consenso absoluto sobre os conceitos relacionados com workflow, existe um esforço para se criar padrões para tal tecnologia. Isso se concretizou com a WfMC (Workflow Management Coalition) [27], um consórcio de empresas desenvolvedoras de tecnologia de workflow, que visa criar e fomentar o uso de padrões de definição e intercomunicação entre as várias ferramentas existentes.

Inicialmente, os sistemas de gerenciamento de workflow limitavam-se a passar os diferentes itens de trabalho entre os respectivos participantes de um processo. Hoje, a tecnologia amadureceu, e o processo como um todo é automatizado, desde a definição de um processo de workflow até o controle de fluxo necessário à execução de cada uma de suas etapas. Com isso, um sistema de workflow não é mais apenas um roteador de tarefas, mas é responsável pelo processo que lhe é requisitado executar.

Mas, mesmo com diferenças de implementação, pode-se notar um forte conceito intuitivo de workflow. Esse conceito está associado à melhoria dos processos, onde a tecnologia de workflow é usada para analisá-los de forma coerente, melhorá-los e controlá-los para se atingir um melhor resultado.

Outros conceitos relacionados serão descritos a seguir.

2.1.1 Processo de Negócios

O processo de negócios está inserido dentro do domínio de uma estrutura organizacional e de uma política de negócios, que visa atingir os objetivos da organização [26].

Um processo de negócios é formado por diversas atividades e pelas especificações de fluxo de dados e controle entre elas [2]. Os processos de negócios geralmente são de longa duração, envolvendo muitos usuários e ferramentas sobre ambientes heterogêneos e distribuídos. Podemos citar alguns exemplos de processo de negócio:

1. A compra de produtos para atualização do estoque de uma loja;
2. A reserva de passagens em empresas aéreas e estadias em hotéis de acordo com um roteiro de viagem;
3. A abertura de uma nova conta em um banco;
4. A instalação de uma linha telefônica por uma companhia telefônica.

As atividades acima podem ser divididas em tarefas menores, de forma que, ao final da execução das tarefas, a atividade está concluída. Por exemplo, o processo 4 pode ser dividido nas seguintes tarefas: preencher um formulário com os dados do cliente, cadastrar o cliente no sistema de controle, verificar disponibilidade de linhas na região do cliente, enviar equipe para instalação da linha e efetuar teste da linha. Após o teste da linha, o usuário já pode utilizá-la, portanto o processo está concluído.

2.1.2 Tarefa

As tarefas são cada um dos passos que devem ser executados para que um processo de negócios seja concluído. Um processo de negócios normalmente contém diversas tarefas, dos mais variados tipos e com diferentes tempos de execução. As tarefas podem ser executadas por humanos, por sistemas de software ou por ambos.

Exemplos de tarefas incluem:

1. Atualizar um registro em uma base de dados;
2. Gerar um recibo;
3. Instalar um dispositivo;
4. Preencher um formulário;
5. Efetuar backup de um conjunto de arquivos;
6. Imprimir um documento;
7. Realizar uma ligação telefônica.

2.1.3 Definição de Processo

Definição de processo é a representação computacional de um determinado processo de negócios [27]. Uma definição de processo especifica toda a estrutura de um processo de negócios:

Tarefas. O conjunto de ações que devem ser desempenhadas para que o processo seja completado.

Responsáveis. As entidades responsáveis pela execução de cada tarefa. As entidades podem ser um aplicativo ou um humano. Pode existir um mesmo responsável para diversas tarefas, bem como diversos responsáveis para uma determinada tarefa.

Seqüência de execução. A seqüência em que as tarefas devem ser executadas. No exemplo de processo mostrado anteriormente, a tarefa “efetuar teste da linha” não pode ser executada antes da tarefa “enviar equipe para instalação da linha”. Então, algum tipo de seqüenciamento se faz necessário.

Fluxo de dados. As informações geradas por uma tarefa podem ser necessárias para tarefas posteriores. Assim sendo, é necessário especificar como essas informações são passadas de tarefa para tarefa.

2.1.4 Instância de processo

Uma instância de processo é uma execução particular de uma determinada definição de processo, com parâmetros e dados próprios.

Podem existir diversas instâncias de uma mesma definição em execução simultaneamente. Por exemplo, para a definição de processo abertura de uma conta em um banco, pode existir uma instância para o cliente X e outra para o cliente Y.

2.1.5 Tipos de SGWfs

A maioria das classificações são baseadas na similaridade entre os processos de negócio envolvidos e seu valor na organização. Outras de acordo com a tecnologia utilizada para fornecer suporte à execução dos processos [15]. Entretanto, é possível classificá-los também de acordo com a complexidade das tarefas e suas estruturas. A Figura 2.1 mostra estas relações.

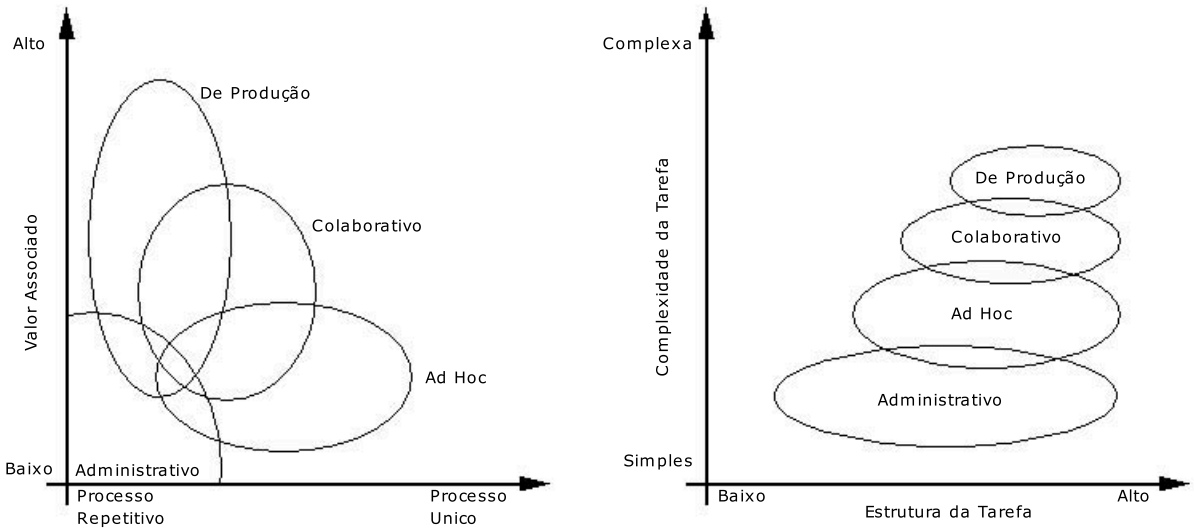

Figura 2.1: Relações entre os tipos de workflow.

A taxonomia mais aceita [3] agrupa os SGWfs de acordo com a estrutura e complexidade dos workflows por ele gerenciados:

- **Administrativos**. Referem-se a processos nos quais as tarefas a serem executadas são bem determinadas e seguem um conjunto de regras conhecido por todos os envolvidos em sua execução. A estrutura desse tipo de workflow tende a não variar muito ao longo do tempo. Exemplos de processos administrativos incluem: registro de um veículo, matrícula de disciplinas em uma universidade e demais processos nos quais existe um conjunto de formulários a serem preenchidos durante a execução de cada tarefa.

- **Ad Hoc**. São similares aos workflows administrativos, com a diferença de que são capazes de lidar com exceções e casos especiais, onde determinada instância pode exigir um tratamento único. Um exemplo é um processo de submissão de artigos para revistas, onde são executados diferentes protocolos dependendo de cada revista.

- **Colaborativos**. Sua característica principal é a presença de um grande número de participantes que colaboram para a execução das tarefas. Ao contrário dos outros tipos de workflow, onde há sempre uma execução seqüencial das tarefas, nos workflows colaborativos podem existir diversas iterações sobre uma mesma tarefa até que um consenso seja atingido. Além disso, podem ocorrer retornos a tarefas anteriores. Um exemplo é a edição de um artigo por diversos autores de forma cooperativa.

- **De Produção**. São caracterizados como a implementação de processos de negócios críticos, ou seja, aqueles processos que estão diretamente relacionados à função da empresa. São workflows administrativos que consideram aspectos como execução em larga escala, alta complexidade dos processos e heterogeneidade de ambiente, além de variedade de participantes envolvidos e tarefas executadas. Por isso tendem a ser executados em grandes empresas, onde esses fatores estão presentes com maior freqüência. Exemplos são controle de pagamentos de seguros e pedidos de empréstimo.

2.1.6 Arquitetura Básica

Os SGWFs, em alto nível, podem ser caracterizados por sistemas que dão apoio a três áreas funcionais conforme Figura 2.2:

- **Funções de definição em tempo de construção** definem, e possivelmente modelam, o workflow e suas tarefas constituintes.

- **Funções de controle em tempo de execução** gerenciam o workflow em um ambiente operacional e fazem o seqüenciamento das várias tarefas que compõem cada processo.

- **Funções de interação com usuários e aplicações em tempo de execução** controlam a interação do usuário com as várias tarefas e gerenciam a invocação de aplicações, com o objetivo de executar os vários passos de uma determinada tarefa.

De acordo com a WfMC [26], pode ser definida uma arquitetura básica a partir da grande variedade de SGWFs no mercado. Esta arquitetura identifica os componentes principais de um SGWF, suas funcionalidades e um conjunto de interfaces entre os mesmos.

Os componentes desta arquitetura (Figura 2.3) são os seguintes:

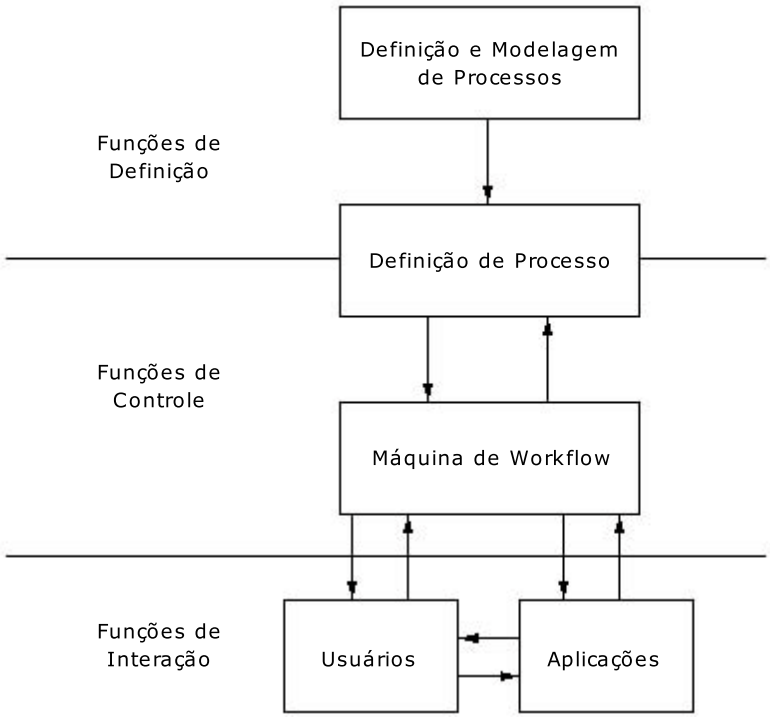

Figura 2.2: Áreas funcionais básicas de SGWfs.

- **Serviço de Execução**. Provê o ambiente de execução para os processos, através de uma ou mais máquinas de workflow. É responsável pela criação, gerenciamento e execução dos processos, seqüenciando a execução das tarefas, invocando as aplicações e controlando a interação com os usuários.

- **Ferramentas de Definição de Processos**. Geram a definição do processo a ser executado. Esta definição contém informações a respeito das atividades que compõem o processo, das regras para a execução das atividades, referências a aplicações que serão invocadas e a definição de qualquer dado relevante que será necessário.

- **Aplicações Cliente de Workflow**. Gerenciam a interação entre os usuários participantes da execução de um processo e o Serviço de Execução. Interagem com os usuários em atividades que necessitam da ação humana, através de uma Lista de Trabalho.

- **Lista de Trabalho**. É uma lista de tarefas atribuidas pela máquina de workflow a um determinado usuário. O usuário pode selecionar itens individuais desta lista, processá-los e informar ao sistema o resultado da execução dos mesmos.

- **Aplicações Invocadas**. A execução de uma ou mais aplicações pode ser necessária para completar a execução de determinadas tarefas.

- **Outros Serviços de Execução**. Diferentes Serviços de Execução podem ter de operar em conjunto para a execução de um mesmo processo.

- **Ferramentas de Administração e Monitoramento**. Provêm serviços para alterar regras de alocação de trabalho, identificação dos participantes que podem desempenhar cada papel dentro de um processo, visualização do histórico de execução de um processo ou atividade, entre outros. Estes serviços são disponíveis a determinados usuários, os chamados supervisores ou administradores.

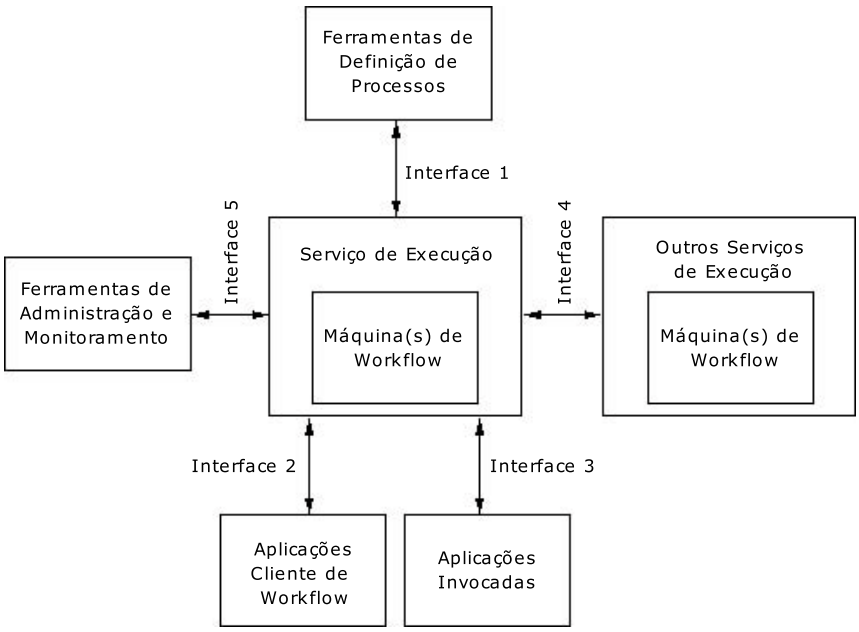

Figura 2.3: Componentes e interfaces da Arquitetura básica de SGWfs.

O Serviço de Execução interage com todos os demais através de cinco interfaces (figura 2.3), as quais definem as formas de comunicação entre cada componente e também como ocorre a transferência de dados e informações entre eles. Isso garante que dois SGWFs construídos respeitando as definições das interfaces serão capazes de interoperar, independentemente de suas implementações.

2.2 Flexibilidade

Processos de negócio dentro de empresas sofrem mudanças freqüentes visando o melhoramento de desempenho e a adequação a novas leis ou regimentos. Para acomodar mudanças, os módulos de especificação e de execução de workflows devem estar estreitamente ligados. Por

exemplo, deve ser possível editar a especificação de um workflow e então, dinamicamente e com segurança, realizar a mudança nas instâncias de processos de workflow em execução [9].

Podemos definir tal sistema de gerenciamento de workflow como “Flexível”.

A flexibilidade em SGWfs é um dos tópicos de pesquisa mais estudados atualmente nessa área. Experiências de modelagem de sistemas reais e complexos na área de pesquisa de mercado [11] trazem à tona a necessidade de flexibilidade. Neste estudo foram identificados vários problemas devidos à complexidade do processo modelado. Entre eles podemos citar:

- impossibilidade de identificar todos os passos e definir o fluxo de controle a priori.
- decisão de quais passos realmente incluir conforme a probabilidade de serem realmente executados.
- possibilidade de erros no mapeamento do mundo real para o modelo de processo.

Neste contexto, a flexibilidade na execução de workflows é essencial. Dois aspectos de flexibilidade devem ser oferecidos por SGWfs:

- **Um usuário deve ter liberdade para escolher entre diferentes fluxos de execução, se necessário**. A linguagem de definição do processo deve prover técnicas que permitam a modelagem de um grande número de fluxos alternativos de execução, de uma maneira compacta.
- **Deve ser possível alterar a instância do processo durante sua execução**. Pode ser que a instância em execução não esteja de acordo com as exigências atuais. O SGWf deve prover técnicas de adaptação destas instâncias de forma que novos requisitos possam ser atendidos.

2.2.1 Classificação

Nesta seção será descrita uma taxonomia [11] que permite classificar os requisitos de flexibilidade em cenários de aplicações reais de forma satisfatória. O termo flexibilidade está relacionado com aplicações de workflow e não SGWf. O requisito de flexibilidade afeta diretamente usuários, projetistas e administradores que lidam com as aplicações de workflow.

Podemos identificar dois tipos de flexibilidade. No primeiro tipo, um workflow pode ter várias interpretações (vários caminhos possíveis) compatíveis com o modelo de processo. Este tipo de flexibilidade é chamada de “Flexibilidade por Seleção”, porque um dos vários fluxos de execução possíveis deve ser selecionado em tempo de execução.

A segunda interpretação de flexibilidade esta ligada à adaptação de modelos de processo ou instâncias de processo. Uma adaptação na instância se torna necessária quando a flexibilidade

por seleção não é mais suficiente para cobrir uma situação em particular. Um exemplo deste fato é a inclusão de tarefas temporárias que possivelmente não ocorrerão novamente em outras instâncias futuras. Em tal caso, a instância deve ser modificada (ou adaptada). Este tipo de flexibilidade será chamada de "Flexibilidade por Adaptação".

Flexibilidade por Seleção

Flexibilidade por Seleção fornece ao usuário um certo grau de liberdade na execução de uma instância, oferecendo múltiplos fluxos alternativos de execução.

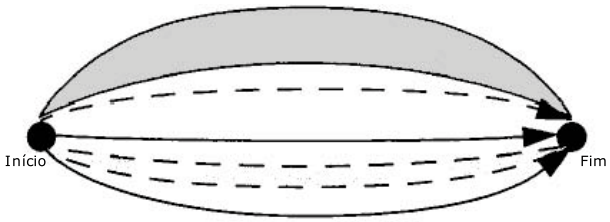

Figura 2.4: Flexibilidade por Seleção.

A Figura 2.4 ilustra o conceito de flexibilidade por seleção. Os pontos de início e fim representam, de maneira abstrata que o objetivo é partir do ponto inicial até ao final da execução de um fluxo qualquer do processo. As linhas contínuas entre estes pontos representam alguns fluxos de execução diretamente (ou explicitamente) especificados no modelo de processo. Já as áreas hachuradas entre estes pontos representam o conjunto de fluxos alternativos de execução que são especificados indiretamente (ou implicitamente) no modelo de processo.

Podemos identificar dois métodos para implementar a flexibilidade por seleção, diferenciados pelo momento em que são aplicados:

- **Modelagem Antecipada**. Feita em tempo de construção.
- **Modelagem Tardia**. Feita em tempo de execução

Na modelagem antecipada, o grau de liberdade provido para o usuário é implicitamente ou explicitamente definido em tempo de construção do modelo de processo. É essencial que a linguagem de especificação do processo forneça formas de modelagem compactas ou pouco complexas, de forma que as especificações do processo continuem legíveis.

Na modelagem tardia, partes do processo são modeladas em tempo de execução. Estas partes são chamadas de "Caixas Pretas". A construção de caixas pretas deve fazer parte da linguagem de especificação do processo. Na Figura 2.4, os fluxos de execução associados com as caixas pretas são ilustrados como linhas tracejadas entre os pontos inicial e final.

Estas caixas pretas são uma espécie de subprocessos definidos em tempo de execução e que fazem parte de um dos fluxos modelados em tempo de construção. Isto é, em tempo de construção, deve-se especificar no modelo de processo onde estão as caixas pretas. Já em tempo de execução, se estas caixas pretas forem ativadas, um subprocesso é definido a partir desta caixa preta.

Resumindo, na flexibilidade por seleção o grau de flexibilidade deve ser previsto em tempo de construção e o apoio à flexibilidade pode ser incluído em um modelo de processo diretamente (Modelagem Antecipada) ou indiretamente (Modelagem Tardia). Uma implementação de flexibilidade bem sucedida é aquela onde a execução de um determinado fluxo nunca irá causar uma situação de exceção.

Flexibilidade por Adaptação

A flexibilidade por seleção é limitada pois deve ser prevista na especificação do processo e deve ser incluída (direta ou indiretamente) na especificação. Em alguns cenários de aplicação isto pode não ser suficiente.

A flexibilidade por adaptação permite a adição de modificações não previstas em tempo de modelagem. A Figura 2.5 demonstra como um dado conjunto de fluxos de execução é estendido por um outro através da adaptação (ou modificação) da especificação do processo ou de suas instâncias. A flecha em negrito representa uma modificação adicionada a uma instância ou modelo de processo. Para prover flexibilidade por adaptação, o SGWf deve fornecer ferramentas adicionais para mudar a definição do processo e integrar estas mudanças em tempo de execução.

Figura 2.5: Flexibilidade por Adaptação.

Foram identificados dois tipos de flexibilidade por adaptação [11]:

- Adaptação da especificação (modelo ou definição) do processo.
- Adaptação de instância.

A adaptação da definição do processo não afeta as instâncias em execução. No segundo caso, deve ser decidido quais instâncias de processo serão afetadas:

- Aquelas instâncias que serão instanciadas no futuro.
- Aquelas instâncias que ainda não atingiram a parte modificada do processo.
- Um conjunto conhecido de instâncias de processo.

No contexto de flexibilidade por adaptação, os termos “versão” e “variante” se tornam mais importantes. Uma nova versão da definição de um processo substitui a versão atual. Só pode existir uma versão válida em um determinado momento. As instâncias de processo somente poderão ser criadas a partir de uma versão válida. Pode ser definido também um período de validade de uma versão e quando esta será substituída por outra.

Entendemos que o termo “versão” diz respeito ao histórico de diferentes definições de processo que possam existir, sendo que apenas uma é válida e pode dar origem a novas instâncias de processo.

Múltiplas variantes de uma definição de processo podem coexistir, ou seja, em um mesmo momento múltiplas variantes de um processo podem ser válidas. Instâncias de processo podem ser criadas a partir de qualquer variante válida.

O termo “variante” difere do termo “versão” no aspecto de validade das definições de processo, ou seja, múltiplas variantes podem ser válidas e podem dar origem a novas instâncias de processo, diferentemente do termo “versão”, onde apenas uma definição é válida em um dado momento.

É uma decisão de projeto do SGWf usar “versões” ou “variantes” de definições de processo. Isto é, pode-se escolher dentre ambas aquela que se encaixe mais perfeitamente nos objetivos do projeto do SGWf. Podemos também optar por usar ambos os conceitos. Neste caso, se uma definição de processo foi alterada, deve ser decidido se essa alteração resultará em uma nova versão ou em uma nova variante. Se for uma nova versão, deve ser decidido quando a mesma se tornará válida.

2.3 Trabalhos Correlatos

Conforme já foi dito, a flexibilidade e adaptabilidade dos sistemas de workflow é um assunto amplamente estudado, onde existem diversas visões e modelagens do problema. Foram escolhidos alguns trabalhos que vão de encontro ao objetivo desta dissertação e portanto serão comentados nesta seção.

2.3.1 Modelo de Liu et. al.

Liu et. al. [16] descrevem o processo de adaptação de um workflow como um processo evolutivo onde políticas de evolução devem ser definidas para que este processo flua normalmente. Estas políticas podem ser executadas em três passos:

1. Modificar a definição de um processo a partir de sua definição atual, criando uma nova que atende as necessidades do processo evolutivo.

2. Especificar uma política de mudança para as instâncias em execução, de maneira que as mesmas façam parte do processo evolutivo.

3. Aplicar esta política de mudança.

Tais passos, serviram de base para a proposta de uma linguagem de especificação para mudanças. Esta linguagem permite definir posições de uma instância em execução e ações a serem tomadas no caso de mudança de especificação. Uma das ações possíveis é o cancelamento semântico de parte ou todo o trabalho executado pela instância. A segunda ação é a de migração da instância em execução para a nova especificação. A terceira ação (default) é ir em frente de acordo com a especificação antiga.

Assim, é possível especificar políticas como:

- instâncias em execução prosseguem de acordo com a especificação antiga, enquanto novas instâncias são iniciadas seguindo a nova especificação.

- instâncias em execução são abortadas.

- instâncias em execução são migradas para a nova especificação. Parte do trabalho realizado pode ter que ser cancelado se corresponder a um caminho afetado pela mudança.

Tais situações podem ser diretamente especificadas usando a linguagem proposta. As operações básicas definidas são: GO-AHEAD, ROLLBACK e CHANGE OVER. Esta operações combinadas suportam quaisquer tipos modificações em situações específicas. Outro detalhe importante é que estas modificações podem ser agendadas quando uma determinada tarefa se torna ativa.

2.3.2 ADEPT flex

ADEPTflex [19] é baseado no modelo formal de workflow ADEPT (Application Development Based on Encapsulated Premodeled Process Templates) [18]. A partir dele, foi desenvolvido um conjunto mínimo e completo de operações de mudança que permitem que os usuários modifiquem a estrutura das instâncias em execução, preservando suas propriedades de corretude. Se uma mudança viola estas propriedades, será rejeitada ou tratada como exceção. Além disso, o sistema dá apoio a mudanças temporárias e permanentes na estrutura das instâncias de processo, prevenindo possíveis problemas no caso de uma mudança temporária precisar ser desfeita.

Duas operações básicas são definidas:

- **Inserção dinâmica de Tarefas**. Tarefas podem ser inseridas dinamicamente no fluxo de execução da instância. Tal inserção deve ser verificada com relação a sua corretude e validade, em caso de falha a operação é abortada. Novas dependências entre tarefas são estabelecidas, para que o processo continue sendo consistente.

- **Exclusão dinâmica de Tarefas**. Tarefas pertencentes a uma instância de processo em execução podem ser excluídas conforme a necessidade, porém alguns aspectos precisam ser verificados. Primeiro, a relação de dependências entre a tarefa a ser excluída com as demais do processo, deve ser verificada para que a consistência do processo seja mantida. Segundo, uma exclusão pode particionar o grafo de dependências, fato este que não pode ser permitido. No caso de ocorrer algum desses problemas a exclusão é abortada.

Reichert e Dadam [19] também definem diretrizes para a aplicação de mudanças temporárias e permanentes. Para realizar estes tipos de mudança, é definido um histórico de mudanças, que serve para armazenar o estado da instância antes da mudança, para que, em caso de falhas, o mesmo possa ser restaurado.

No caso das mudanças temporárias, verificações de consistência e corretude são realizadas. Em caso de falha, a mudança é abortada. Caso contrário, a mudança é realizada e o estado anterior à mudança é armazenado no histórico de mudanças.

No caso de mudanças permanentes, verificações adicionais devem ser realizadas antes que a mudança seja efetuada. Pois, uma mudança permanente feita de maneira errada pode causar falhas irrecuperáveis na execução de processos. Além disso, a aplicação de uma mudança permanente pode afetar outras mudanças temporárias feitas previamente.

2.3.3 Trabalho de Aalst e Jablonski

Aalst e Jablonski [1] classificam as mudanças de acordo com seis critérios:

1. **A razão da mudança**. Neste critério, são consideradas as mudanças externas ao processo de negócio e e as mudanças internas. As mudanças externas acontecem no ambiente onde o processo está implantado e podem ocorrer em três circunstâncias: Mudança do contexto do negócio; Mudança de leis; Mudanças tecnológicas. Já nas mudanças internas os problemas são detectados no processo em si e podem acontecer em duas circunstâncias: Erros lógicos no projeto de negócio e Problemas técnicos.

2. **O efeito da mudança**. De acordo com a natureza da mudança (externa ou interna) o efeito por ela causado pode ser temporário ou evolutivo. Efeitos temporários não demandam mudanças na definição do processo, pois são mudanças que ocorrem por eventos raros no processo, exceções ou uma demanda especial do cliente. O grande dilema é decidir

quais mudanças serão permitidas, pois pode ser que se torne inviável realizá-las temporáriamente. Os efeitos evolutivos são de natureza estrutural e ocorrem em decorrência de mudanças de estratégia no negócio, mudanças de leis e reengenharia de processo. Estas mudanças são tipicamente iniciadas pelos gerentes de negócio ou forçadas por novas leis ou novos ambientes externos.

3. **As perspectivas afetadas**. São cinco as perspectivas afetadas: Perspectiva do processo; Perspectiva da organização; Perspectiva da informação; Perspectiva de operação; Perspectiva de integração. Tais perspectivas vão desde mudanças em tarefas do processo (adição, exclusão, etc.), passando por mudanças nos papéis e pessoal da organização, mudanças nas estruturas de dados e documentos utilizados no processo, mudanças em aplicações utilizadas no processo e, finalmente, mudanças na integração de todas as perspectivas, por exemplo, a ligação de um novo papel a uma nova tarefa.

4. **O tipo de mudança**. As mudanças podem ser classificadas em quatro tipos: Extensão; Redução; Alteração; Religação. As mudanças do tipo extensão ocorrem na inserção de uma nova tarefa, papel ou novo usuário dentro do processo. As mudanças do tipo redução são o inverso da extensão, ou seja, ocorrem na exclusão de entidades do processo. As mudanças do tipo alteração são um misto entre a redução e a extensão, por exemplo, a troca de uma tarefa X por outra Y dentro do processo. Finalmente, nas mudanças do tipo religação ocorrem após qualquer uma das outras mudanças mencionadas anteriormente. Por exemplo, a reorganização das dependências entre as tarefas após uma redução no processo.

5. **Quando as mudanças são permitidas**. As mudanças podem ser permitidas em duas circunstâncias: Antes do início do processo; Durante a execução do processo. No primeiro caso, a mudança deve ser aplicada antes do início do processo, não sendo permitidas novas alterações a partir do momento que o processo entra em execução. No segundo caso, as mudanças são permitidas enquanto a instância do processo está em execução e no momento que se achar oportuno.

6. **O que fazer com as instâncias em execução**. Diversas atitudes podem ser tomadas em relação às instâncias em execução. Primeiro, as instâncias antigas são descartadas, e novas instâncias, já de acordo com as mudanças, entram em execução. Segundo, as instâncias antigas são abortadas e reinciadas de acordo com a nova especificação do workflow. Terceiro, somente novas instâncias são executadas de acordo com a nova especificação. As instâncias em execução não tem seu fluxo de execução alterado. Quarto, as instâncias em execução tem seu fluxo de execução alterado de acordo com a nova especificação. Finalmente, para mudanças temporárias pode ser feito um desvio de fluxo até que o evento que gerou a mudança seja tratado.

Além disso, o artigo propõe uma forma de lidar com mudanças em processos utilizando herança. Na perspectiva de processo, uma classe corresponde à definição de processo e os objetos (i.e. instâncias de uma classe) correspondem às instâncias de processo. Porém, como o conceito de herança tradicional é restrito à estrutura de uma classe, novas formas de herança são apresentadas como mais adequadas para lidar com o comportamento dinâmico.

2.3.4 Trabalho de Bogia e Kaplan

Bogia e Kaplan [6] apresentam um modelo para lidar com mudanças em definições de processo baseado em herança múltipla. Cada tarefa tem dois tipos de especificação: especificação geral, que define como uma tarefa deve ser executada; especificação local, resultado de modificações nas especificações gerais.

As modificações globais causam mudanças em todas as tarefas iniciadas a partir de um conjunto de especificações gerais. A partir dessas modificações, novas versões se tornam disponíveis e os usuários podem optar por adotar a nova versão se houver compatibilidade. No caso de incompatibilidade ou exceções, modificações locais podem ser necessárias.

A implementação do modelo proposto é chamada de *Obligations Network* (Rede de Obrigações) onde as obrigações são passadas entre os agentes participantes do processo. Juntamente com essas obrigações são passadas informações de contexto, prioridade, tempo de expiração da obrigação, uma descrição do objetivo a ser atingido, bem como uma rede de sub-obrigações que devem atendidas para completar a execução da obrigação. A rede de obrigações é interconectada por ligações de entrada e saída, onde são passadas fichas (semelhantes às das Redes de Petri) que representam o estágio atual de execução das obrigações.

A rede de obrigações é composta por múltiplas camadas:

- Camada de informações de instância;
- Camada de modificações locais.
- Camada de especificações gerais.

Na camada de informações de instância são armazenadas informações sobre o estado atual da rede, tais como: quais ações estão ativas, quais foram completadas e quantas vezes cada objeto da rede foi invocado. Desta forma, esta camada provê um registro histórico do ciclo de vida de todos os objetos invocados e que estão em execução.

A camada de modificações locais contém um conjunto de modelos que representam todas as mudanças que foram feitas na rede para tornar uma obrigação de acordo com a mudança requerida. Estas mudanças não afetam nenhuma outra obrigação que tenha passado por uma mudança similar.

A camada de especificações gerais contém um conjunto compartilhado de modelos que representam o processo global a ser seguido. Qualquer mudança realizada neste conjunto afeta toda a rede de obrigações que compartilham o mesmo. Quando este processo global não é conhecido, é criado um modelo que define somente um ponto de início e um ponto de chegada para o processo.

Sendo assim, podemos observar algumas vantagens deste modelo:

- Existe uma clara distinção entre as especificações gerais do processo e as modificações locais;
- Múltiplos modelos de especificação geral permitem aos participantes do processo usar inúmeras filosofias de herança para a especificação de seu processo. Isto permite que inúmeras filosofias similares possam ser criadas e utilizadas de acordo com a necessidade;
- Múltiplos modelos de modificação local permitem uma série de níveis de controle aos participantes do processo. Primeiro, em cada modelo são definidas permissões de execução e que tipos de mudança podem ser executados (inserção, exclusão, etc.). Segundo, através da hierarquia de modelos, um participante pode controlar a partir de que nível da mesma uma mudança deve ser realizada;
- Através da utilização de herança a complexidade na especificação do processo é reduzida, pois a tarefa se reduz a criar uma hierarquia de herança especificando novas informações a cada nível criado;
- Como as mudanças locais estão em uma camada diferente das especificações gerais, fica simples identificar eventos que ocorrem com frequência e que possam vir a gerar um novo modelo na camada de especificação geral;
- Como as informações de instância estão em uma camada específica, a extração de informações de controle de obrigações passadas para reuso fica simples. Desta forma, qualquer obrigação, seja em execução ou completada pode ser usada como modelo para instanciação ou ser armazenada em um arquivo de modelos.

2.3.5 Trabalho de Sadiq et. al.

Sadiq et. al. [23] definem dois aspectos de especificação do processo: o modelo de processo de workflow e o fluxo de controle e execução de workflow. O modelo de processo define a lógica de negócio para instâncias particulares. Já o fluxo de controle e execução consiste em um conjunto de itens de trabalho que contém informações relativas à execução de atividades dentro do processo.

Tendo como base esta definição, as principais características propostas são:

- Introdução de uma camada entre a definição e a execução do workflow. Esta representa uma camada de livre acesso à especificação do processo para uma instância em particular;
- Ter uma especificação de processo que somente constitua uma definição parcial do processo.

Na especificação do processo são pré-definidas atividades estáticas e atividades especiais de construção que permitem a definição de subprocessos em tempo de execução. Assim, uma definição de processo consiste de um núcleo de um ou mais pontos de flexibilidade definidos em tempo de execução. No momento em que esta atividade é executada, a instância pode se tornar uma instância padrão (ou uma nova versão) do processo. Desta forma, a cada nova instância criada, novas versões podem existir, tornando a execução dos processos totalmente flexível.

Em resumo, o artigo apresenta um modelo capaz de capturar a lógica de processos flexíveis complexos sem comprometer a simplicidade e generalidade da linguagem de definição de processos. Isto é realizado através de pontos de flexibilidade inseridos nas definições do processo que permitem que o processo seja definido por completo em tempo de execução e de acordo com a necessidade.

2.3.6 ML-DEWS

Por fim, ML-DEWS [9] é uma linguagem visual projetada para especificar e construir os artefatos de um processo, bem como as mudanças que nele serão realizadas. A ML-DEWS é uma extensão da UML (Unified Modeling Language) [24]. Esta linguagem é baseada em um pequeno número de conceitos proposto pela WfMC [26]. Todos os elementos do modelo de workflow tais como processos, atividades, regras de dependência, eventos e fluxos são modelados como classes. Resumindo, um processo é um objeto cujo comportamento é parcialmente descrito por seu fluxo de controle e parte pelo seu fluxo de dados.

O processo de mudança implementado por uma classe de mudança de processo que descreve o processo de mudança para todas as entidades participantes do mesmo, de maneira simples, para que as mesmas se comuniquem e deleguem as responsabilidades. Esta classe contém informações sobre quando a mudança será iniciada e qual o tempo de expiração. Existem, ainda, o filtro de mudanças, que especifica o subconjunto de casos antigos que migrarão para o novo processo, e o processo de migração, que encapsula os tipos de mudanças que serão aplicadas aos casos antigos. Cada processo de migração está associado a somente uma classe de mudança de processo.

2.3.7 Correlação com o presente trabalho

Basicamente, o presente trabalho usou como inspiração os trabalhos de Reichert e Dadam [19] e Sadiq et. al. [23], no que diz respeito a incorporação de operações para fornecer flexibilidade

por adaptação e por seleção.

A flexibilidade por adaptação do presente trabalho, tem grande semelhança com o conjunto de operações de mudança proposto por Reichert e Dadam [19] que permitem que os usuários modifiquem a estrutura das instâncias em execução.

Já a flexibilidade por seleção, tem grande semelhança com o trabalho de Sadiq et. al. [23], no que diz respeito às tarefas de construção de subprocessos, onde na especificação do processo são pré-definidas atividades estáticas e atividades especiais de construção que permitem a definição de subprocessos em tempo de execução. Assim, uma definição de processo consiste de um núcleo de um ou mais pontos de flexibilidade definidos em tempo de execução.

Os outros trabalhos citados tem correlação com a teoria de flexibilidade em workflows, tais como o de Liu et. al. [16] e Aalst e Jablonski [1]. Já os trabalhos de Bogia e Kaplan [6] e Ellis e Keddara[9], tratam do assunto apresentando outras soluções para o problema de flexibilidade.

Capítulo 3

Sistema WorkToDo

O WorkToDo [21][22] é um sistema de gerenciamento de workflows distribuído que utiliza a camada ORB-CORBA [7] como plataforma de comunicação. Seu objetivo principal é permitir a execução de processos de workflow em ambientes de comunicação sem fio. Usuários podem executar tarefas sem estar conectados à máquina de workflow ou conectados através de uma rede sem fio.

O WorkToDo é um SGWf com a arquitetura distribuída, de acordo com os aspectos discutidos na seção 2.1 do capítulo 2.

Nesse capítulo, enfocaremos os aspectos principais do sistema sem considerar as funcionalidade voltadas à operação desconectada. Serão apresentados o modelo de processo na seção 3.1, a arquitetura na seção 3.2 e as funcionalidades básicas em 3.3. Na seção 3.4, serão apresentados os aspectos da implementação do protótipo. Esse protótipo foi usado como base para a implementação das operações de flexibilidade discutidas no capítulo 4.

3.1 Modelo de Processo

O modelo de processo no WorkToDo inclui especificações para os seguintes aspectos [14]:

- **Aspecto Funcional**. Define que ações devem ser executadas pelo processo sendo que um processo consiste de um conjunto de atividades.
- **Aspecto Comportamental**. Define as interdependências entre as diversas atividades do processo, de maneira que se possa saber qual atividade será executada em um dado momento.
- **Aspecto Organizacional**. Define quem será o responsável pela execução de uma determinada atividade. Este responsável pode ser um humano ou uma aplicação.

- **Aspecto Informacional**. Define quais serão os dados utilizados por cada atividade, além do fluxo dos mesmos entre as atividades.

No WorkToDo, uma instância de processo é criada a partir de uma definição de processo, consistindo de um conjunto de atividades e dependências entre elas. Cada atividade possui um conjunto de dados de entrada e saída. Uma atividade pode ser uma tarefa ou conjunto de tarefas executadas por entidades processadoras. A figura 3.1 mostra a representação do modelo na notação UML (Unified Modeling Language) [24].

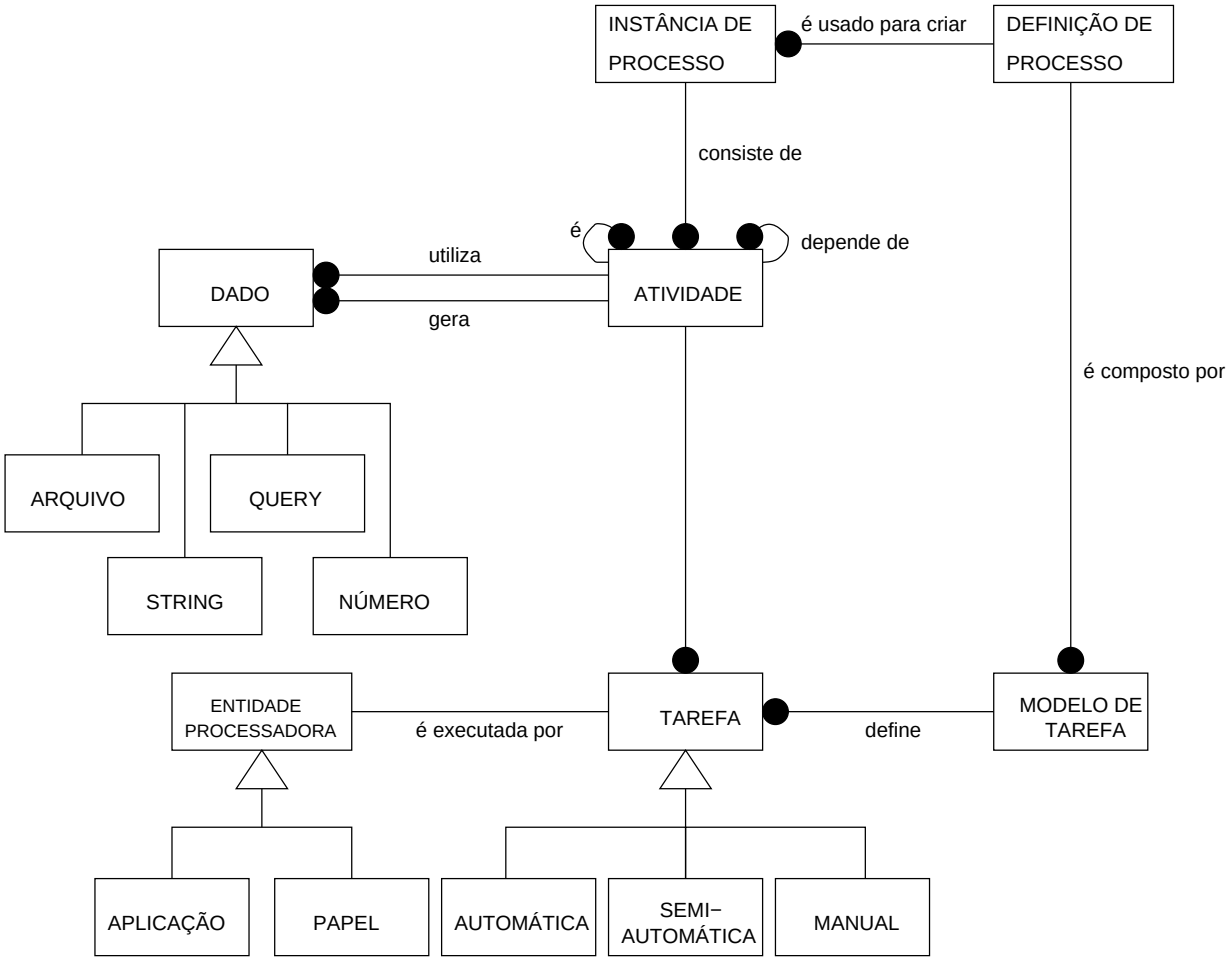

Figura 3.1: Modelo de processo do WorkToDo em UML.

3.1.1 Atividades

As atividades constituem o aspecto funcional de um processo. Uma atividade consiste de uma ou mais tarefas. As tarefas são unidades básicas de um processo. No contexto desse trabalho,

utilizaremos somente o conceito de tarefa definido a seguir.

3.1.2 Tarefas

As tarefas são unidades básicas de um processo como, por exemplo, a edição de um arquivo texto ou o envio de um e-mail. Uma tarefa é definida por um Modelo de Tarefa (figura 3.1) que contém o tipo da tarefa, sua prioridade e o papel de quem a deve executar.

As tarefas podem diferir umas das outras, exigindo tratamentos distintos por parte do sistema. Por isso elas são classificadas em três tipos:

Automáticas. São executadas por algum software invocado automaticamente pelo SGWf (por exemplo: acessar um banco de dados, copiar um arquivo).

Semi-automáticas. São executadas por um humano auxiliado por um sistema de software (por exemplo: redigir um documento em um editor de textos).

Manuais. São executadas exclusivamente por um humano (por exemplo: preencher um formulário, enviar uma carta pelo correio).

Estados das Tarefas

Em um dado momento, uma tarefa pode estar em um dos cinco estados abaixo:

NOT_READY - As condições para a execução da tarefa não foram satisfeitas.

READY - As condições para a execução da tarefa foram satisfeitas. Essas condições são avaliadas usando-se uma árvore de dependências explicada em 3.1.3.

RUNNING - A tarefa foi selecionada (escalonada) para execução.

SUCCEEDED - A tarefa foi terminada com sucesso.

FAILED - A tarefa passa para o estado FAILED se houver falha durante sua execução.

As transições entre os estados de uma tarefa são mostradas na figura 3.2. O estado inicial de uma tarefa é o estado NOT_READY. Já os estados finais são: SUCCEEDED e FAILED.

3.1.3 Regras de Dependência

Para modelar o aspecto comportamental de um processo, o WorkToDo utiliza regras de dependência. Uma regra de dependência contém as condições que devem ser satisfeitas para que determinada tarefa seja executada. Estas condições são descritas em termos de transições

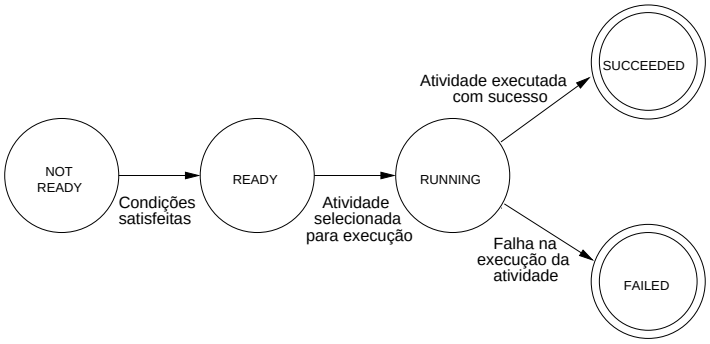

Figura 3.2: Diagrama de transição de estados de uma tarefa.

de tarefas para estados. Uma regra é composta por zero ou mais termos, onde cada termo é formado por um nome de tarefa e um estado. Os termos podem se relacionar através dos operadores booleanos `and` e `or`. Um exemplo de regra de dependência é mostrado abaixo:

$$\text{and } (T_1 \rightarrow \text{SUCCEEDED}, T_2 \rightarrow \text{SUCCEEDED})$$

Nesse exemplo, a tarefa associada à regra somente será executada se as tarefas T_1 e T_2 alcançarem o estado `SUCCEEDED`. A regra de dependência possui dois termos: $T_1 \rightarrow$ `SUCCEEDED` e $T_2 \rightarrow$ `SUCCEEDED`, relacionados através da operação `and`. Os estados válidos para as regras são aqueles possíveis para as tarefas, conforme a Seção 3.1.2. Uma regra de dependência vazia significa que a tarefa não depende de nenhuma condição para ser executada.

3.1.4 Entidades Processadoras

Para toda tarefa existe uma ***Entidade Processadora*** responsável por sua execução. As entidades processadoras modelam o aspecto organizacional de um processo e são definidas nos modelos de tarefas. O WorkToDo define dois tipos de entidades processadoras:

Aplicações. São as entidades processadoras das tarefas automáticas. Para executar a tarefa, a aplicação correspondente deve ser invocada (possivelmente com parâmetros e/ou dados de entrada/saída). A cada tarefa automática é atribuída uma aplicação.

Usuários. São as entidades processadoras de tarefas semi-automáticas ou manuais. A atribuição de tarefas aos usuários se dá através de um sistema de papéis. Um ***Papel*** designa um grupo de usuários que possuem determinadas características. Na especificação da tarefa é definido um papel R, de maneira que qualquer usuário pertencente a R pode executar a tarefa.

3.1.5 Dados

As tarefas podem utilizar dados de entrada e gerar dados de saída, dados estes que correspondem ao aspecto informacional dos processos. No WorkToDo, tais dados podem ser de quatro tipos: Arquivos – arquivos armazenados em disco; *Queries* – expressões em SQL (*Simple Query Language*) que definem consultas a bancos de dados e que possivelmente retornam um conjunto de dados (registros); Strings – valores alfanuméricos; e Números – valores inteiros ou reais.

Cada instância de processo utiliza uma área exclusiva, denominada ***Contexto do Processo***, na qual armazena seus dados. Assim, quando uma tarefa necessita recuperar ou atualizar dados, o faz nessa área. O contexto de uma instância de processo P somente é acessível às tarefas pertencentes a P, isolando assim os dados de cada instância de processo e evitando que a execução de um processo interfira em outra.

3.1.6 Linguagem de Definição

O WorkToDo utiliza uma ***Linguagem de Definição de Processo – LDP*** (apêndice A), através da qual é possível especificar definições de processos, modelos de tarefas e aplicações. Uma definição de processo contém uma lista de tarefas e, para cada tarefa, seus dados de entrada e saída, suas dependências e sua entidade processadora. Um modelo de tarefa especifica as características da tarefa, como tipo e prioridade. Já a definição de uma aplicação contém as características da aplicação, como tamanho e sistema operacional no qual pode ser executada.

3.2 Arquitetura

No WorkToDo os componentes estão dispostos em uma arquitetura distribuída. São os seguintes (figura 3.3):

- **Gerenciador de Usuários e Papéis (User and Role Manager - URM)**. Gerencia os usuários, os papéis existentes e a lista de usuários pertencentes a cada papel.
- **Gerenciador de Definições (Definition Manager - DM)**. Armazena e controla o acesso às definições de processo, modelos de tarefas e aplicações.
- **Gerenciador de Instâncias (Instance Manager - IM)**. Armazena informações sobre as instâncias de processo em execução e também as que já foram encerradas.
- **Gerenciador de Processo (Process Manager - PM)**. Controla a execução de uma determinada instância de processo.
- **Gerenciador de Tarefa (Task Manager - TM)**. Controla a execução de uma determinada tarefa automática.

- **Gerenciador de Lista de Trabalho (Worklist Manager - WM)**. Gerencia uma lista de tarefas (semi-automática e manuais) que podem ser executadas por um determinado usuário.

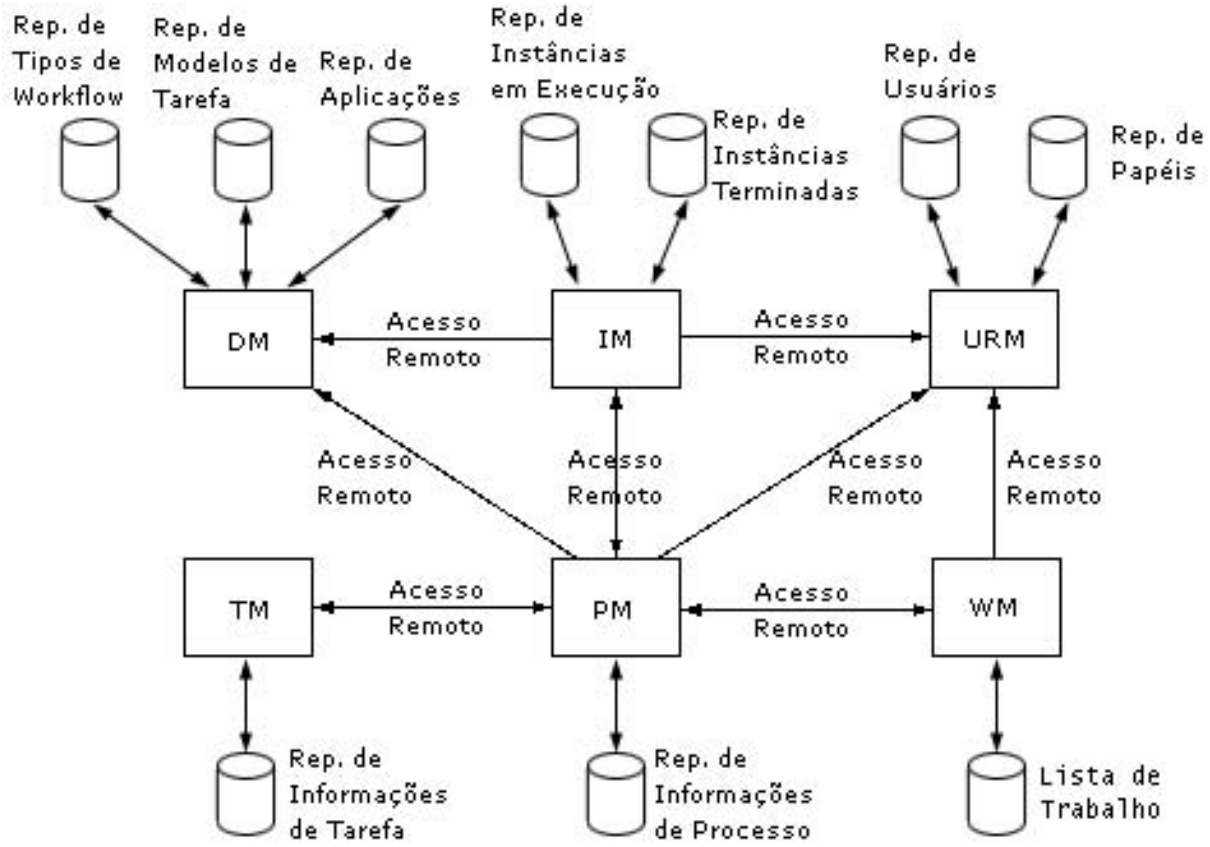

Figura 3.3: Arquitetura do WorkToDo.

A arquitetura distribuída torna o sistema mais tolerante a falhas, já que no caso de uma falha, apenas os componentes na máquina que falhou se tornarão inacessíveis, enquanto o restante do sistema continua funcionando normalmente. Outra vantagem está no fato de não haver carga excessiva em uma máquina, evitando sobrecarga de processamento e gargalo na comunicação.

As seções a seguir descrevem com mais detalhes cada um dos componentes desta arquitetura.

3.2.1 Gerenciador de Usuários e Papéis

O Gerenciador de Usuários e Papéis (User and Role Manager - URM) é responsável por gerenciar quais usuários têm acesso ao sistema, os papéis definidos dentro de uma organização e a

lista de usuários para cada papel. Cada vez que um componente do sistema deseja acessar ou modificar informações a respeito dos usuários e/ou papéis, esta requisição é feita ao URM, que consulta o repositório correspondente.

Este componente controla os seguintes repositórios:

- **Usuários**. Armazena informações a respeito dos usuários do sistema, tais como: nome, estado da conexão, endereço IP de sua máquina e uma lista de mensagens recebidas.
- **Papéis**. Armazena os papéis existentes e a lista de usuários dentro de cada papel.

3.2.2 Gerenciador de Definições

O Gerenciador de Definições (Definition Manager - DM) é responsável por gerenciar o acesso às definições de processo, modelos de tarefas e aplicações. Permite recuperar as mesmas, além de adicionar novas definições ou removê-las. Desta forma, controla três repositórios que armazenam as definições de processos, modelos de tarefas e os modelos de aplicação.

Cada vez que um componente do sistema necessita de informações sobre o processo, tarefas e/ou aplicações, esta requisição é feita ao DM, que consulta o repositório correspondente. Quando uma nova definição de processo é criada, o DM requisita o nome de um papel, chamado Papel Criador. Sendo assim, somente o usuário com esse papel pode criar instâncias deste processo.

3.2.3 Gerenciador de Instâncias

O Gerenciador de Instâncias (Instance Manager - IM) é responsável por armazenar o estado dos processos em execução e também o estado final dos processos encerrados. O IM também é responsável pela criação de um Gerenciador de Processos descrito em 3.2.4 para cada nova instância de processo.

Para prover estas informações, o IM controla um repositório de processos, que contém informações sobre o estado de cada processo em execução, e um repositório de processos terminados, que armazena informações sobre esses processos.

3.2.4 Gerenciador de Processo

O Gerenciador de Processo (Process Manager - PM) é o componente responsável por coordenar a execução de uma instância de processo. Ele verifica quais tarefas estão prontas para executar, inicia sua execução e coleta seus resultados, verificando se tais resultados permitem a execução de novas tarefas.

A cada PM, ou seja, a cada instância de processo, está associada uma lista de tarefas que contém a definição de todas as tarefas pertencentes ao processo. Informações como contexto de

entrada e saída, árvore de dependência, tarefas dependentes, estado atual e usuário que selecionou a tarefa, são armazenadas numa estrutura chamada *definição de tarefa*.

Na criação de uma instância de processo, é construída uma lista de tarefas a partir de uma especificação na Linguagem de Definição de Processo - LDP. O PM interpreta essa especificação e constrói a lista de tarefas correspondente.

O PM é composto por dois sub-componentes descritos a seguir:

- **Escalonador**. Tem por função avaliar as condições para a execução das tarefas, de acordo com sua árvore de dependências. Isto é, se uma tarefa T sofre uma transição para um estado E, o escalonador avalia a árvore de dependências de todas as tarefas que dependem da transição de T para o estado E. Quando uma árvore de dependência é avaliada para TRUE, o estado da tarefa passa para READY.

- **Despachante**. Prepara as tarefas para a execução. No caso de uma tarefa automática, o despachante cria um Gerenciador de Tarefa para controlar a execução da tarefa. Já no caso de uma tarefa semi-automática ou manual, o despachante notifica todos os usuários que pertencem ao papel da tarefa a respeito de sua disponibilidade.

3.2.5 Gerenciador de Tarefa

O Gerenciador de Tarefa (Task Manager - TM) é criado por um PM e invoca a aplicação responsável pela execução de uma determinada tarefa automática. No momento da criação do TM, o PM repassa a definição de tarefa correspondente. Após a aplicação iniciar sua execução, o TM a monitora e, quando ela é finalizada, notifica o PM que o criou, informando o tipo de término (sucesso ou falha).

Se a execução de uma tarefa falha, o TM verifica se o limite de repetições da tarefa foi alcançado. Se não foi, a aplicação é novamente invocada. Somente quando o número de repetições é ultrapassado o TM notifica o PM correspondente a respeito da falha da tarefa e manda uma mensagem para o usuário responsável informando sobre a falha ocorrida.

3.2.6 Gerenciador de Lista de Trabalho

O Gerenciador de Lista de Trabalho (Worklist Manager - WM) tem por função manter uma lista de tarefas que podem ser executadas por um determinado usuário, ou seja, a cada usuário existe um WM responsável por controlar sua Lista de Trabalho. Cada tarefa pertencente a esta lista é denominada Item de Trabalho.

Existe uma relação entre o Item de Trabalho e sua respectiva tarefa, ou seja, a cada alteração de estado do item, a tarefa correspondente no processo também sofre alteração de estado, garantindo assim que as alterações efetuadas pelos usuários se reflitam nos processos.

3.2.7 WorkToDo vs. Modelo de Referência

A arquitetura do WorkToDo é compatível com o modelo de referência apresentado na Seção 2.1.6. Esta relação se dá da seguinte forma:

- O PM, IM, DM e URM correspondem ao Serviço de Execução, controlando e gerenciando a execução das instâncias de processos e suas respectivas tarefas;
- As Aplicações Cliente de Workflow são controladas no WorkToDo pelos WMs;
- As Aplicações Invocadas são controladas pelo WM, no caso de tarefas semi-automáticas, ou pelo TM, no caso de tarefas automáticas;
- As Ferramentas de Definição de Processos são implementadas pela LDP, armazenadas pelo DM e interpretadas pelo PM no momento da criação do processo;
- As Ferramentas de Administração e Monitoramento estão implementadas em uma interface específica para o Administrador do processo, onde podem ser realizadas operações pertinentes ao processo, aos usuários, aos papéis e às definições do processo;
- A característica de interoperabilidade com outros SGWfs não foi implementada.

O WorkToDo por ser capaz de gerenciar diversos tipos de processos, pode ser classificado como um SGWf Administrativo e/ou de Produção.

3.3 Funcionalidades Básicas

Nesta Seção, são descritas as funcionalidades básicas do WorkToDo. Isto é, quais são os tipos de usuários existentes (Seção 3.3.1), como é a interação dos mesmos com o sistema (Seção 3.3.2), notificação e prazo de tarefas (Seção 3.3.3 e Seção 3.3.4), os tipos de mensagens enviadas no sistema (Seção 3.3.5) e, finalmente, os diagramas de sequência de algumas operações básicas (Seção 3.3.6).

3.3.1 Tipos de Usuários

Os usuários do WorkToDo podem ainda assumir os seguintes papéis [20]:

- **Administrador**. É o papel assumido pou um ou mais usuários, que permite que se tenha alguns privilégios com relação aos demais usuários. Tais privilégios são: Gerenciamento de usuários e papéis, Gerenciamento de Definições de Workflow e Consulta a estado dos processos.

- **Participante**. É o papel assumido pelos usuários que participam da execução de uma instância de processo. Os papéis e os usuários pertencentes a um papel são definidos pelo usuário administrador.

- **Criador ou responsável pelo processo**. É o usuário que cria uma instância de processo. O WorkToDo assume automaticamente o usuário criador da instância como sendo o usuário responsável pelo processo. Esse usuário deve acompanhar o andamento do processo e tomar certas decisões quando for necessário. Também recebe notificações de prazos e falhas de tarefas e sobre o início e o término do processo pelo qual ele é responsável.

  Este usuário também é capaz de realizar as operações de flexibilidade que serão descritas posteriormente neste trabalho.

3.3.2 Interação com os Usuários

Os usuários do sistemas efetuam a conexão e as operações do SGWf através de uma interface, que fornece mecanismos de comunicação com os componentes do sistema. A interação do usuário com o sistema ocorre principalmente por meio de sua lista de trabalho, através da qual o usuário visualiza quais tarefas estão prontas para a execução e por onde pode realizar as seguintes operações:

- **Visualizar itens de trabalho**. Todas as tarefas pertencentes ao usuário são exibidas de forma ordenada: por ordem de chegada, prioridade ou por prazo.

- **Selecionar itens de trabalho**. A seleção ocorre quando o usuário deseja executar uma tarefa. Ao selecionar um iten de trabalho (somente para tarefas manuais ou semi-automáticas), o PM correspondente é notificado e remove o iten de trabalho das demais listas de tarefas dos usuários daquele papel, avisando seus respectivos WMs. O WM do usuário recebe então uma confirmação e, a partir daí, o usuário pode executar a tarefa selecionada. Particularmente em tarefas semi-automáticas, feita a seleção, o WM do usuário invoca a aplicação correspondente à tarefa. Terminada a execução da aplicação o usuário informa o resultado da tarefa ao sistema.

- **Informar o resultado da execução de um iten de trabalho**. No caso de tarefas manuais, o resultado (SUCCEEDED ou FAILED) da execução das tarefas selecionadas por usuário é informado por esta operação.

A interface do WorkToDo também permite ao usuário:

- Verificar quais são os processos em execução.

- Consultar o estados dos processos dos quais participa.

- Consultar processos terminados.
- Criar instâncias a partir de uma definição de processo, desde que o usuário pertença ao papel criador dessa definição.

3.3.3 Notificação de Tarefas

Essa funcionalidade permite que o WorkToDo notifique os usuários conectados ao SGWf que existem tarefas prontas para executar. No momento em que uma tarefa fica pronta (passa para o estado READY), o Despachante notifica o WM de todos os usuários pertencentes ao papel da tarefa.

3.3.4 Prazos para Tarefas

No WorkToDo, a cada tarefa é associado um prazo para que a mesma seja executada por um usuário. Este prazo é especificado na definição do modelo de tarefa e representa o tempo máximo para que uma tarefa seja executada. Quando esse prazo está se aproximando, o PM envia mensagens periódicas ao usuário responsável e ao usuário que selecionou a tarefa. Se o prazo se esgotou sem que a tarefa seja executada, o usuário responsável é notificado e decide se a execução do processo deve continuar ou não.

3.3.5 Mensagens

No WorkToDo, cada usuário do sistema tem a ele associada uma caixa de mensagens, contendo notificações relevantes para cada ocorrência de eventos. O Gerenciador de Usuários e Papéis (URM) é quem mantém esta lista.

Exitem quatro tipos de mensagens de notificação enviadas no WorkToDo:

- **Início de Processo**. Mensagem de notificação ao usuário responsável pelo processo indicando início do processo.
- **Término de Processo**. Mensagem de notificação ao usuário responsável pelo processo indicando término do processo.
- **Falha de Tarefa**. Mensagem de notificação ao usuário responsável pelo processo indicando que uma determinada tarefa falhou. O responsável pode então identificar qual o problema que causou a falha e tomar as medidas que julgar convenientes.
- **Prazo de Tarefa**. Mensagem de notificação ao usuário responsável pelo processo, bem como o usuário que está com a tarefa selecionada (caso houver algum), indicando que o prazo para a execução da tarefa está se esgotando.

3.3.6 Diagramas de Seqüência

Esta seção apresenta os diagramas de seqüência das principais operações do WorkToDo, com o objetivo de tornar mais claro o funcionamento do sistema e discutir alguns detalhes importantes.

As operações sempre se iniciam por meio de uma requisição do usuário à interface do sistema, que então encaminha a requisição ao componente adequado. Ao final, a interface retorna uma notificação ao usuário, indicando o sucesso ou falha da operação.

Criação de uma Instância de Processo

O procedimento para criação de uma instância, mostrado na Figura 3.4, é detalhado a seguir:

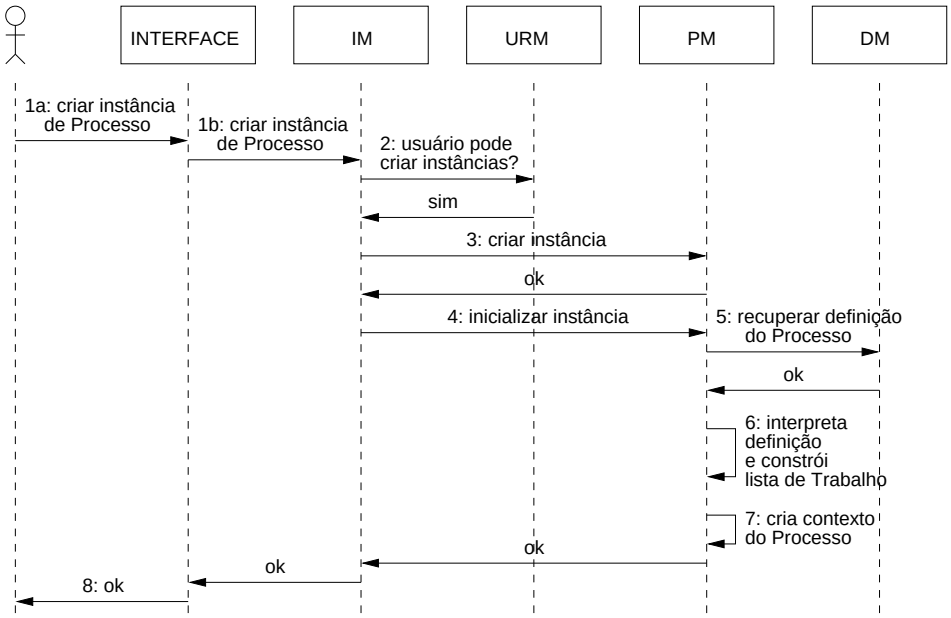

Figura 3.4: Diagrama de seqüência para a criação de uma instância de processo.

1. Através da interface do sistema, o usuário requisita ao IM que seja criada uma instância de processo, informando a definição de processo e a máquina a partir do qual a instância será gerenciada;
2. O IM verifica, junto ao URM, se o usuário pertence ao papel criador daquela definição de processo;
3. O IM cria um novo PM na máquina especificada pelo usuário;
4. O IM inicializa a instância, ou seja, informa ao PM recém-criado qual o tipo de processo que este irá gerenciar;
5. O PM recupera junto ao DM a definição do tipo de processo;

6. O PM ativa o analisador, que interpreta a definição do tipo de processo da seguinte forma. Para cada tarefa instanciada na definição, o analisador interpreta a definição do modelo de tarefa correspondente e constrói uma definição de tarefa. Ao final, todas as definições de tarefa são inseridas em uma lista, formando a lista de tarefas do processo;
7. O PM cria o contexto do processo na máquina onde está sendo executado. O PM cria um diretório com o nome da instância, onde são armazenadas as informações relativas à instância (como por exemplo seu estado atual). Os dados utilizados pelas tarefas também são copiados para esse diretório à medida que são necessários;
8. A interface notifica o usuário de que a instância foi criada com sucesso e que sua execução já foi iniciada.

Se qualquer um dos itens entre 2 e 7 falha, a criação da instância é automaticamente cancelada e a interface informa o usuário a respeito da falha na operação. As ações que porventura tenham sido efetuadas (construção da lista de tarefas, criação do contexto) são desfeitas.

Seleção de um item de trabalho

O procedimento para seleção de um item de trabalho, mostrado na Figura 3.5, é detalhado a seguir:

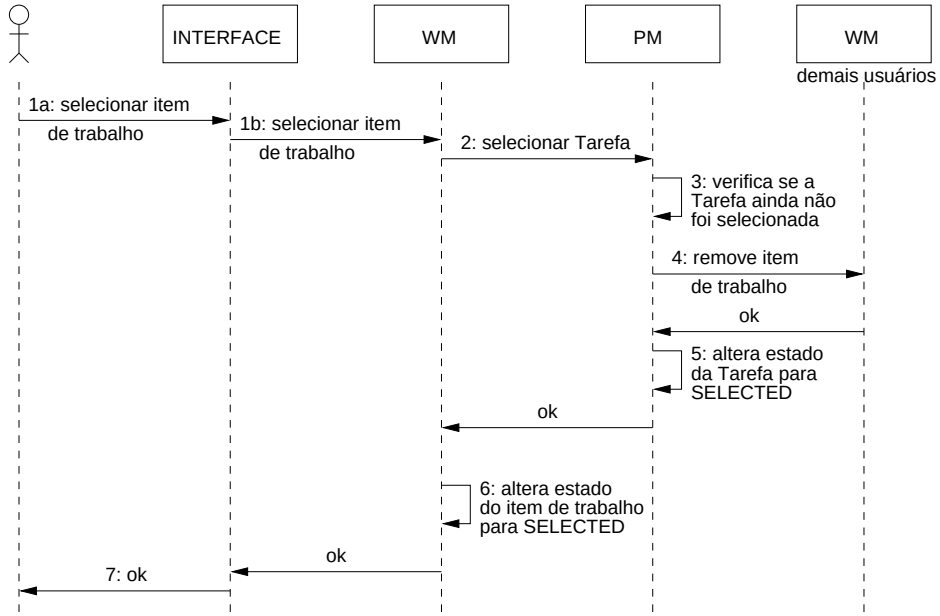

Figura 3.5: Diagrama de seqüência para a seleção de um item de trabalho.

1. Através da interface do sistema, o usuário requisita ao WM a seleção de um determinado item de trabalho;

2. O WM verifica qual a tarefa correspondente ao item de trabalho e envia uma requisição ao PM ao qual a tarefa pertence solicitando a seleção da tarefa para execução;
3. O PM verifica se a tarefa ainda está disponível (algum outro usuário pode tê-la selecionado);
4. O PM remove os item de trabalho correspondentes à tarefa das listas de trabalho dos demais usuários que haviam sido notificados da disponibilidade da tarefa;
5. O PM altera o estado da tarefa para `SELECTED`, indicando que ela foi selecionada por um usuário. Além disso, armazena na definição da tarefa o nome do usuário que efetuou a seleção;
6. O WM altera o estado do item de trabalho para `SELECTED`;
7. A interface notifica o usuário de que o item de trabalho foi selecionado com sucesso e está pronto para ser executado.

Se um dos itens entre 2 e 3 falha, a seleção do item de trabalho é automaticamente cancelada e a interface informa o usuário a respeito da falha na operação.

3.4 Implementação do WorkToDo - OrbixWeb

O protótipo WorkTodo - OrbixWeb [20] foi usado como base para o início da implementação deste trabalho. O protótipo foi implementado em Java - JDK 1.1.8 [13] e a plataforma de distribução utilizada foi o OrbixWeb 3.1c [17], que é um ORB - CORBA distribuído comercialmente pela IONA [12]. O OrbixWeb possui um compilador IDL com mapeamento para Java e a comunicação remota é efetuada através do padrão IIOP. O OrbixWeb, atualmente, encontra-se na versão 6.2c.

O OrbixWeb estende as funcionalidades básicas do ORB, oferecendo serviços de persistência, multi-threading, filtros para operações, localização de objetos por nome, tempo de ativação dos servidores, modos de ativação dos servidores e smart proxies.

Com relação ao protótipo, alguns serviços do OrbixWeb foram utilizados:

- **Sevidores com múltiplas threads**. Minimiza o tempo de resposta ao cliente pelo servidor.

- **Tempo de ativação dos Servidores**. Economiza memória, já que se um servidor fica ocioso, ele é desativado, liberando memória.

- **Persistência de servidores**. Utilizado para salvar o estado dos servidores periodicamente, para ser utilizado em uma reativação.

3.4.1 Pacotes do Protótipo

A figura 3.6 mostra o diagrama de pacotes do WorkToDo. As classes do pacote sysInterface fazem chamadas de métodos das classes pMgr, urMgr, dMgr, iMgr, tMgr e wMgr que implementam os componentes Gerenciador de Processo, Gerenciador de Usuários e Papéis, Gerenciador de Definições, Gerenciador de Instâncias, Gerenciador de Tarefas e Gerenciador de Lista de Trabalho respectivamente.

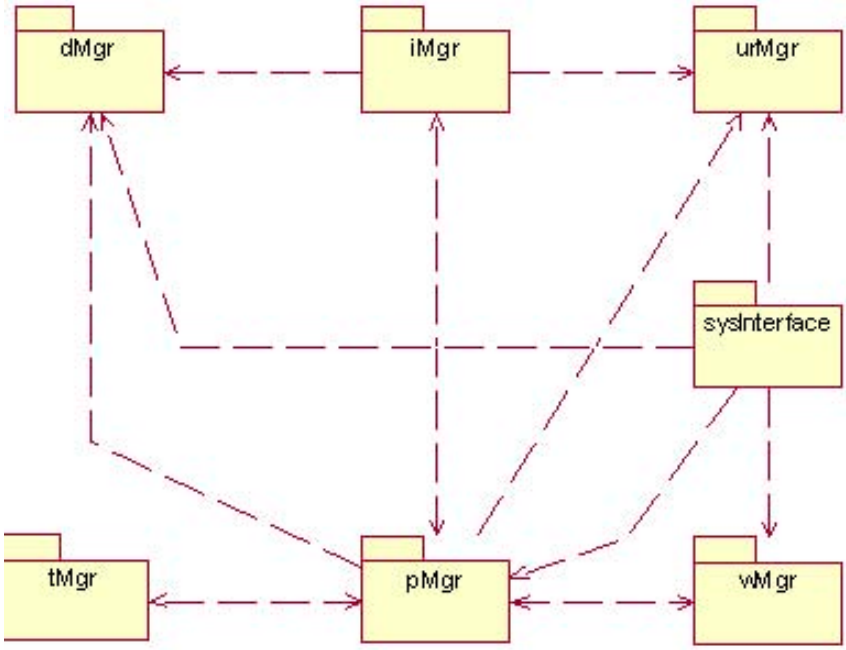

Figura 3.6: Diagrama de Pacotes do WorkToDo.

O pacote dMgr implementa os repositórios de modelos de tarefa, definições de processo e aplicação através das classes TaskModelInfo, ApplicationInfo e WFTypeInfo, além de fornecer uma interface para comunicação remota com outros componentes da arquitetura.

O pacote iMgr implementa os repositórios de processos em execução e processos terminados através da classe ProcessInfo, além de fornecer uma interface para comunicação remota com outros componentes da arquitetura.

O pacote urMgr implementa os repositórios de usuários e papéis através das classes UserInfo e RoleInfo, além de fornecer uma interface para comunicação remota com outros componentes da arquitetura. As mensagens recebidas pelos usuários são armazendas em seu repositório através das classes MessageList e MessageInfo.

O pacote tMgr implementa o serviço de execução para tarefas automáticas através da classe TaskExecutor, além de fornecer interfaces para comunicação remota com outros componentes da arquitetura.

O pacote pMgr implementa, basicamente, três funcionalidades do sistema de gerenciamento de workflow. A primeira, é o analisador responsável por interpretar as especificações e estabelecer as relações de dependências entre as tarefas. Esta funcionalidade é implementada pelo pacote parser.

A segunda é responsável pela criação e monitoramento dos processos, bem como notificação de usuários, monitoramento de prazos de tarefas e execução de tarefas automáticas. Esta funcionalidade é implementada pelas classes Process, ActiveUsersManager, Dispacher, TaskManagerChecker e DeadlineManager.

A terceira é responsável pela criação da lista de tarefas do processo e relações de dependência entre as tarefas da mesma, feita a partir da interpretação realizada pelo analisador. Esta funcionalidade é implementada pela classe TaskListBuilder. Além disso, o pacote pMgr fornece interfaces para comunicação remota com outros componentes da arquitetura.

O pacote sysInterface implementa classes utilizadas nas aplicações dos usuários e que implementam a janela principal do sistema (Classe MainWindow) e janelas secundárias usadas em operações dentro da aplicação (Classes SimpleQueryDialog, ComboBoxDialog, InformatioDialog, OptionsWindow e TextTable). É a partir da classe MainWindow que são requisitadas todas as operações implementadas no sistema de gerenciamento de workflows.

Capítulo 4

WorkToDo Flex - Extensões para Workflow Dinâmico

O WorkToDo Flex é uma extensão do WorkToDo [21][22] que provê um conjunto de operações para alterações dinâmicas nas definições de processo. Neste capítulo serão descritas as mudanças realizadas no WorkToDo para atender aos requisitos de um sistema de gerenciamento de workflows flexível, bem como as operações de flexibilidade implementadas.

Deve ser ressaltado que o WorkToDo Flex foi baseado no WorkToDo, porém adotou-se o RMI (Remote Method Invocation) como meio de comunicação entre os componentes do sistema. As razões para isso são a alta portabilidade da linguagem Java [13] e o fato de que a plataforma Java-RMI oferece os requisitos suficientes para o desenvolvimento do protótipo e exige menos recursos de memória que outras plataformas como, por exemplo, ORB's - CORBA.

Foram alterados com relação ao protótipo WorkToDo - OrbixWeb, os estados das tarefas discutidos na seção 4.1 e a arquitetura com a inclusão de um novo componente chamado Gerenciador de Sub-Processos (apresentada na seção 4.3). Na seção 4.2, são apresentadas as novas operações incluídas para oferecer flexibilidade. Na seção 4.5, são apresentados os diagramas de sequência das operações de flexibilidade. Os diagramas de classes dos componentes que sofreram alterações com relação ao WorkToDo - OrbixWeb são mostrados na seção 4.6.

4.1 Estados das Tarefas

Para implementar as operações de flexibilidade foram criados novos estados para as tarefas (na figura 4.1):

- **READY_SUSPENDED**. Indica que uma tarefa está pronta para ser executada, porém a execução da mesma foi suspensa. Quando uma tarefa passa para este estado, a mesma é removida das listas de trabalho dos usuários;

- **CANCELLED**. Indica que uma tarefa foi cancelada. Quando uma tarefa passa para este estado, a mesma é removida das listas de trabalho dos usuários e todas as tarefas dependentes ficam inacessíveis a não ser que seus estados sejam alterados manualmente (seção 4.2.1);
- **REMOVED**. Indica que uma tarefa foi removida do fluxo de execução da instância de processo em execução. Quando uma tarefa passa para este estado, a mesma é removida das listas de trabalho dos usuários.

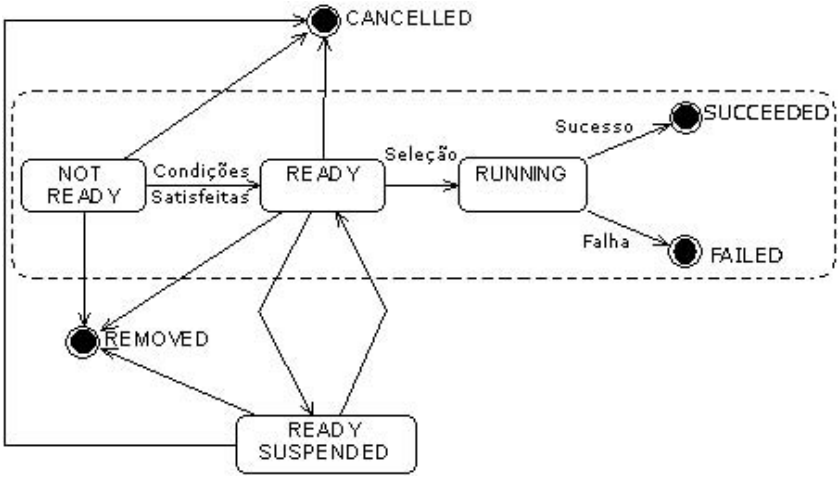

Figura 4.1: Diagrama de estado das tarefas no WorkToDo Flex.

Na Figura 4.1, é apresentado um diagrama de estado das tarefas, onde a área circulada por linhas tracejadas indica os estados originais da implementação do WorkToDo. Os estados fora desta área são os estados adicionados para fornecer as operações de flexibilidade do WorkToDo Flex.

4.2 Flexibilidade no WorkToDo Flex

O WorkToDo Flex provê os dois aspectos de flexibilidade discutidos no capítulo 2: por seleção e por adaptação.

A flexibilidade por seleção é fornecida através da modelagem tardia, ou seja, feita em tempo de execução. Esta modelagem é implementada através de tarefas de construção de subworkflows, ou o que podemos chamar de “caixas pretas” dentro da definição do processo. Sendo assim, a definição do processo prevê pontos onde subworkflows serão definidos e executados durante a execução de uma instância. Tal conceito é semelhante ao proposto por Sadiq et. al. [23]. O funcionamento deste tipo de tarefa no WorkToDo Flex é apresentado na seção 4.2.2.

A flexibilidade por adaptação é fornecida por um conjunto de operações, através das quais o usuário do sistema pode realizar as mudanças desejadas sobre uma instância de processo. Estas operações são descritas com maior detalhe na seção 4.2.1. Estas mudanças são aplicadas em uma instância em execução escolhida pelo usuário responsável. Portanto, a adaptação não é realizada na definição do processo, e sim, com nas instâncias em execução. Após a execução das operações, podemos dizer que uma nova variante do processo foi gerada, porém a partir da mesma não é possível gerar novas versões e também não se tem um mecanismo de validação desta nova variante.

Outro tipo de adaptação não fornecida pelo WorkToDo Flex é a adaptação de tipo. Neste tipo de adaptação deve-se decidir quais as instâncias serão afetadas, pois a mudança é realizada na definição do processo, e a mesma pode gerar uma nova versão. O mecanismo de versões não foi incluído para delimitar o escopo da dissertação.

A seguir serão apresentadas as operações para oferecer flexibilidade de adaptação na seção 4.2.1 e a tarefa de construção em 4.2.2 relacionada com a flexibilidade por seleção.

4.2.1 Flexibilidade por Adaptação

As operações desenvolvidas para oferecer flexibilidade por adaptação são descritas a seguir. Em todas as operações com exceção as de Suspender Workflow e Reiniciar Workflow, é executado o processo de suspensão da instância, para garantir que as mudanças sejam aplicadas ao processo em um estado “congelado”. Além disso a execução de todas as operações é mutuamente exclusiva. Os exemplos apresentados se baseiam no processo de workflow da figura 4.2.

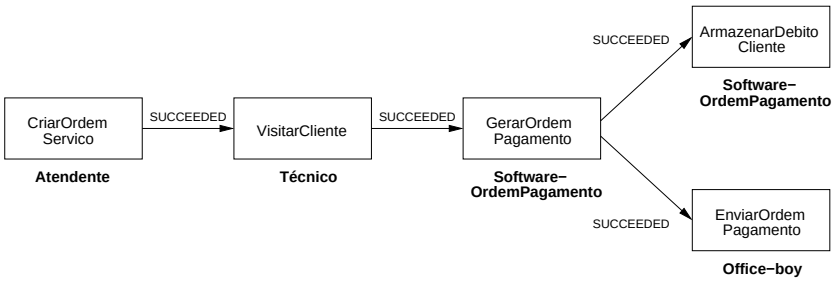

Figura 4.2: Diagrama do processo exemplo.

Cancelar Workflow

O objetivo desta operação é cancelar a execução de uma determinada instância de processo, não importando em qual ponto do fluxo de execução a mesma se encontre. Quando tal operação é requisitada, todas as tarefas que se encontram nos estados NOT_READY, READY e READY_SUSPENDED passam para o estado CANCELLED. As tarefas já executadas e em

execução permanecem nesses estados. A Figura 4.3 apresenta uma tela do protótipo implementado que mostra o estado de uma instância antes e depois de seu cancelamento.

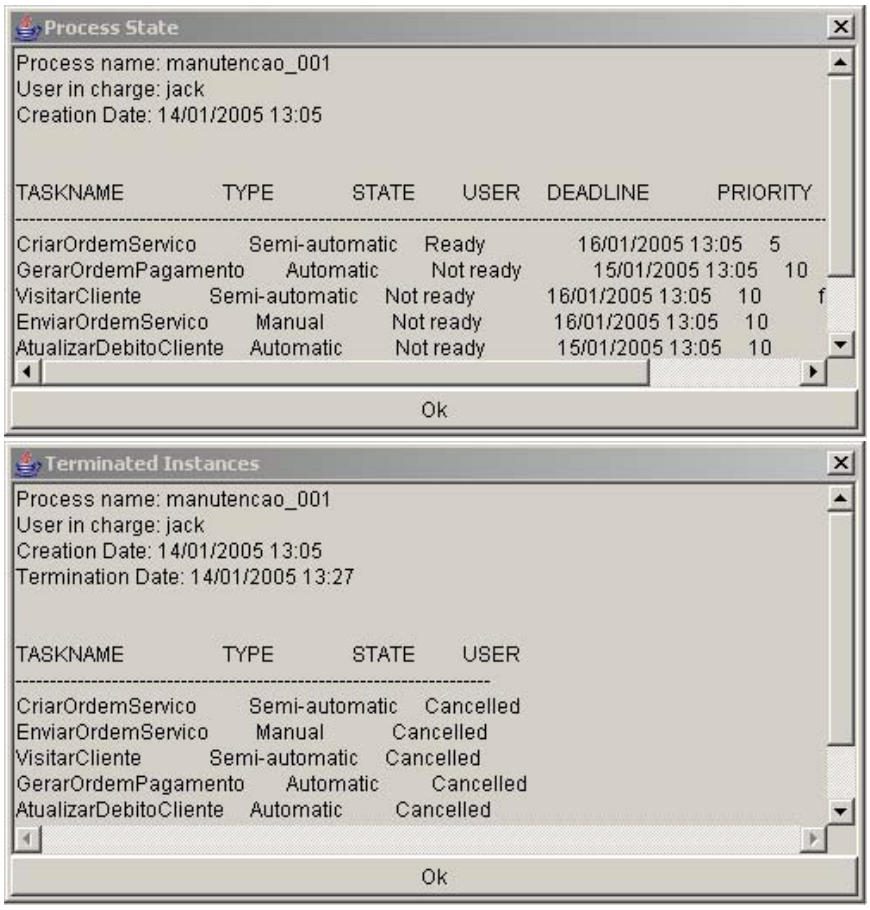

Figura 4.3: Estado de uma instância antes e depois de seu cancelamento.

Portanto, se não há nenhuma tarefa em execução no momento do cancelamento, a instância em execução é encerrada. Caso alguma tarefa ainda esteja em execução, a instância é encerrada após o término de todas as tarefas em execução. Após o cancelamento todos os usuários envolvidos na execução dessa instância são notificados e as tarefas pertencentes à instância cancelada são retiradas de suas listas de trabalho.

Suspender Workflow

O objetivo desta operação é suspender temporariamente a execução de uma determinada instância, não importando em qual ponto do fluxo de execução a mesma se encontre. Quando tal operação é requisitada, todas as tarefas que se encontram no estado READY passam para o estado READY_SUSPENDED. As tarefas já executadas e em execução permanecem em seus

estados atuais. A Figura 4.4 apresenta uma tela do protótipo implementado que mostra o estado de uma instância antes e depois de suspender sua execução.

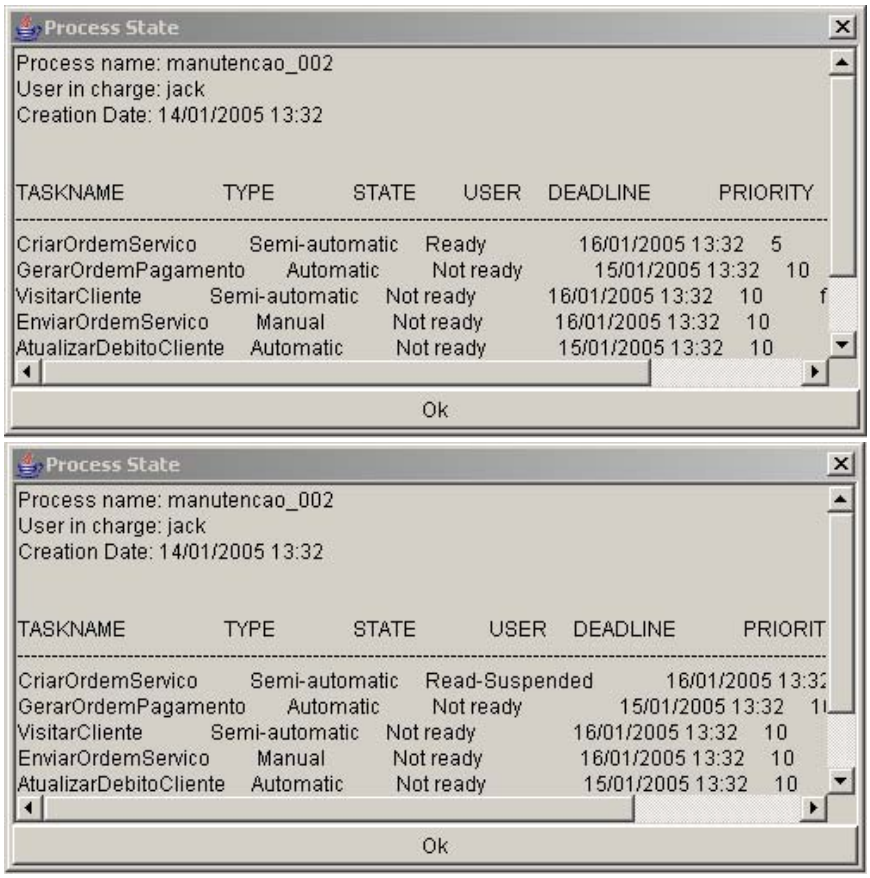

Figura 4.4: Estado de uma instância antes e depois de suspender sua execução.

Portanto, se não há nenhuma tarefa em execução no momento da suspensão, a instância em execução fica “congelada”, isto é, nenhuma outra tarefa pode ser selecionada para execução até que a instância seja reiniciada. Somente as tarefas que já estavam em execução antes da operação ser executada continuam sua seu ciclo normal. Sendo que quando essas terminam, suas tarefas dependentes passam do estado NOT_READY para READY_SUSPENDED, se o processo ainda está suspendido. Caso o processo já tenha sido reiniciado, o fluxo de execução segue normalmente.

Após a suspensão, todos os usuários envolvidos na execução dessa instância são notificados e as tarefas pertencentes à instância suspensa são retiradas de suas listas de trabalho.

Reiniciar Workflow

O objetivo desta operação é reiniciar a execução de uma instância suspensa. Quando tal operação é requisitada, todas as tarefas que se encontram no estado READY_SUSPENDED passam para o estado READY. As tarefas já executadas e em execução permanecem em seus estados atuais. A Figura 4.5 apresenta uma tela do protótipo implementado que mostra o estado da instância exemplo antes e depois de reiniciar sua execução.

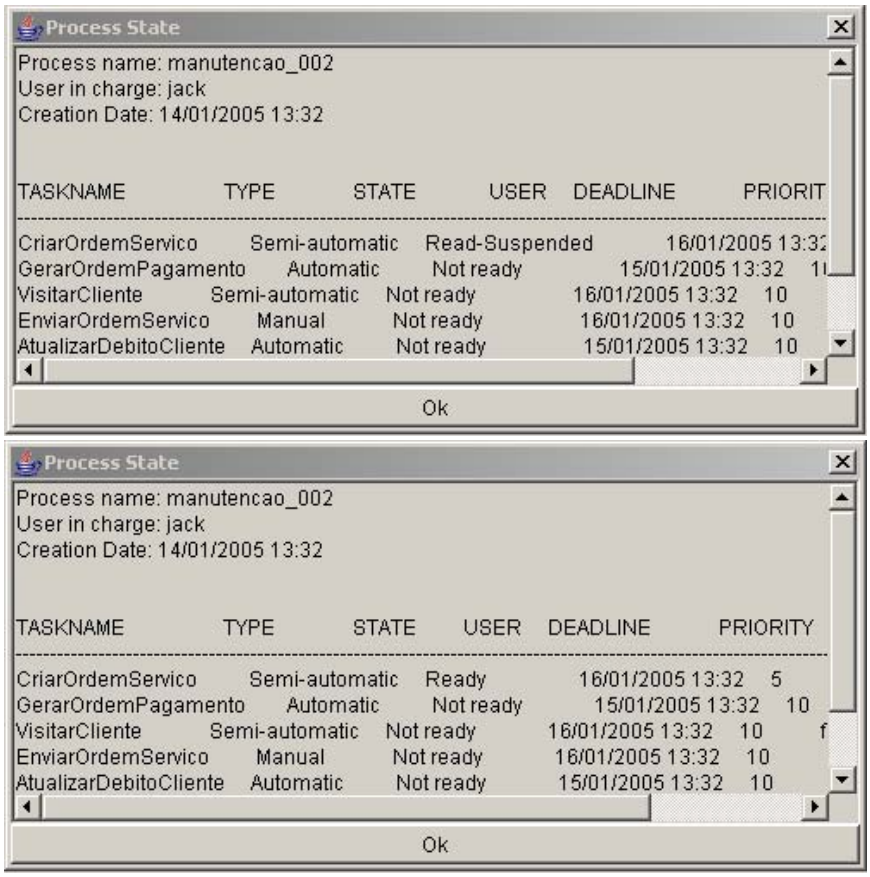

Figura 4.5: Estado de uma instância antes e depois de reiniciar sua execução.

Um detalhe importante, é que uma tarefa só pode passar do estado READY_SUSPENDED para o estado READY se esta não depende de nenhuma outra anterior, ou se a anterior já terminou sua execução, ou seja, está no estado SUCCESS.

Após a execução desta operação, todos os usuários envolvidos na sua execução são notificados e novas tarefas dessa instância podem ser inseridas em suas listas de trabalho.

Inserir nova Tarefa

A adição de uma nova tarefa durante a execução de uma instância se torna necessária em algumas situações. Seja para modelar novos eventos do processo, melhoramentos no mesmo ou até para corrigir possíveis erros na especificação.

Podemos comparar a adição de uma nova tarefa a algo como a adição de um novo procedimento ou função em um algoritmo durante a sua execução. Quando uma nova tarefa é inserida no fluxo de execução, novas dependências também são inseridas para manter a consistência da instância. A Figura 4.6 mostra esta operação, onde as linhas tracejadas representam as novas relações de dependência criadas pela inserção da nova tarefa.

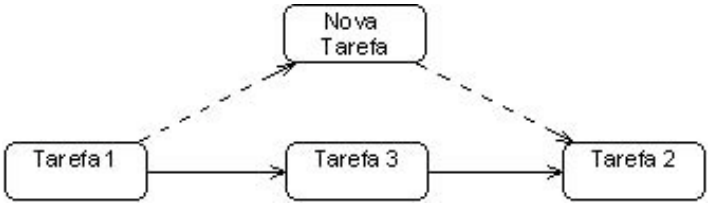

Figura 4.6: Inserção de uma nova tarefa em uma instância.

No WorkToDo Flex, novas tarefas podem ser adicionadas ao fluxo de execução de uma instância, juntamente com suas dependências. Estas novas tarefas são baseadas nos modelos de tarefa já existentes no Gerenciador de Definições e sua árvore de dependência inclui somente as tarefas existentes na definição atual do processo. Porém, outras tarefas que venham a depender desta nova devem ter sua árvore de dependências alterada, ou seja, deve ser executada a operação de alteração de dependências.

A figura 4.7 mostra a execução desta operação no protótipo implementado. No exemplo, é inserida uma nova tarefa T1 cujo modelo de tarefa é definido como ***Preencher Formulário*** e a mesma dependerá do sucesso da tarefa ***Visitar Cliente***. Podem ser escolhidos também arquivos de entrada e saída que serão usados na execução da tarefa.

Após a nova tarefa ser inserida, os usuários do papel apropriado serão notificados quando a mesma estiver pronta para execução. A Figura 4.8 apresenta uma tela do protótipo implementado que mostra o estado da instância exemplo após a inserção da nova tarefa.

Remover Tarefa

Tarefas podem ser removidas do fluxo de execução de uma instância. No WorkToDo Flex somente usuários administradores ou o usuário que criou a instância podem realizar esta operação.

A remoção de uma tarefa X de uma instância em execução somente pode ser executada se X está no estado READY, NOT_READY ou READY_SUSPENDED. Após a remoção de X, a tarefa passa para o estado REMOVED e é retirada das listas de trabalho dos usuários que a

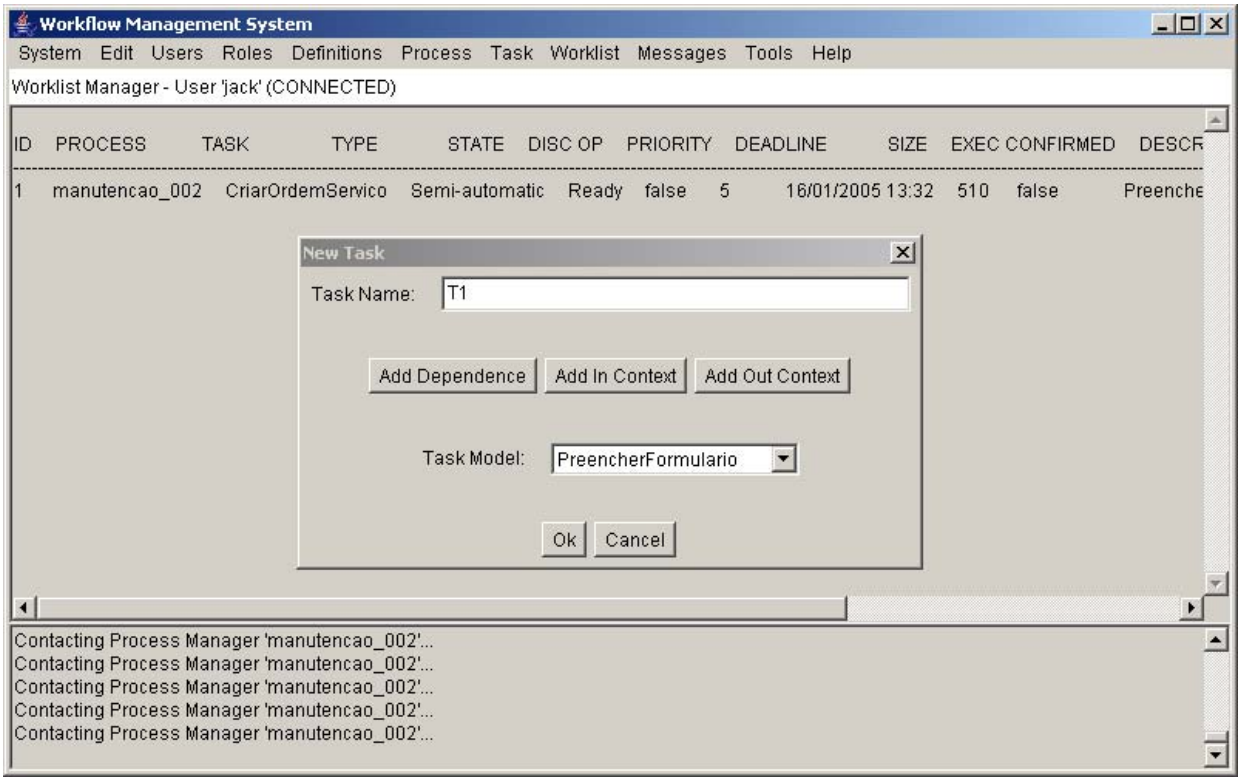

Figura 4.7: Inserção de uma nova tarefa T1.

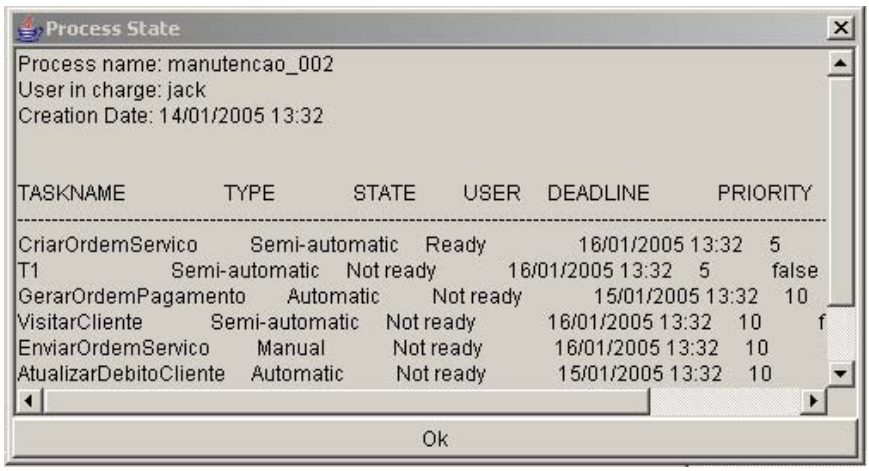

Figura 4.8: Estado de uma instância após a inserção de T1.

receberam. Tarefas no estado RUNNING, SUCCEEDED, FAILED, CANCELLED não podem ser removidas.

Todas as tarefas que dependiam exclusivamente de X passam para o estado READY. Caso

contrário, permanecem no estado atual. A Figura 4.9 mostra o estado da instância exemplo após a remoção da tarefa ***Visitar Cliente***.

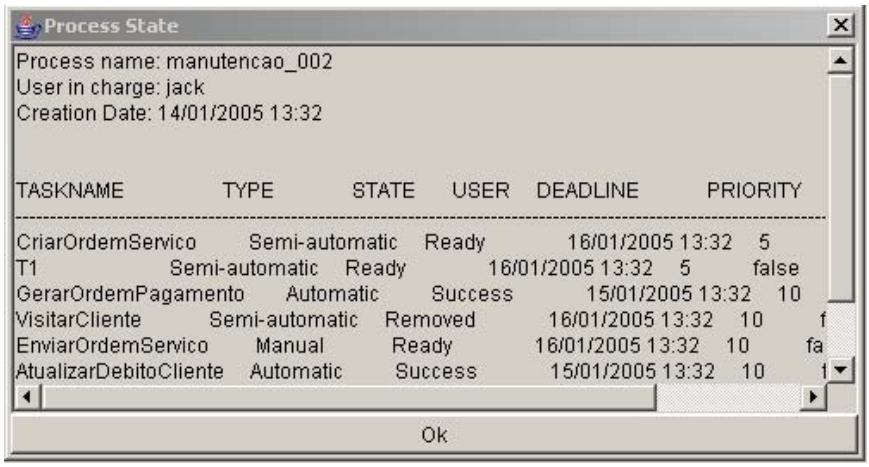

Figura 4.9: Estado de uma instância após a remoção da tarefa Visitar Cliente.

Podemos verificar na figura que as tarefas automáticas ***Gerar Ordem Pagamento*** e ***Atualizar Debito Cliente***, que eram dependentes de ***Visitar Cliente***, tornaram-se prontas e foram executadas instantaneamente por serem automáticas. A tarefa T1, inserida anteriormente, também tornou-se pronta e todos os usuários do papel correspondente foram notificados.

Alterar Dependências de Tarefa

No caso de uma inserção de tarefa, ou ainda, no caso de erro na especificação ou qualquer outra situação inesperada, o WorkToDo Flex fornece uma operação para alteração das dependências de uma tarefa que esteja no estado NOT_READY.

No momento que necessitar, o usuário responsável seleciona a tarefa a ser modificada, podendo assim inserir novas dependências para a tarefa, bem como remover as existentes, criando uma nova árvore de dependências para a tarefa selecionada.

Por exemplo, seja uma tarefa X dependente das tarefas Y e Z. Para que X deixe de ser dependente de Z e passe a depender de uma outra tarefa W, duas operações devem ser realizadas. Primeiro, na lista de dependentes de Z a tarefa X deve ser removida. Por fim, a tarefa X é adicionada na lista de tarefas dependentes de W. Esta operação é ilustrada na Figura 4.10.

No WorkToDo Flex, o usuário responsável escolhe a tarefa a ser alterada e em seguida é exibida uma janela pela qual pode escolher quais tarefas serão removidas e/ou inseridas na árvore de dependência da tarefa selecionada. Em seguida, o sistema avalia a nova árvore para verificar se a tarefa pode mudar para o estado READY ou não. No caso positivo, os usuários que receberam a tarefa na sua lista de trabalho são notificados e sua lista é atualizada com inclusão da tarefa em questão. Um exemplo desta operação no protótipo é ilustrada na Figura 4.11.

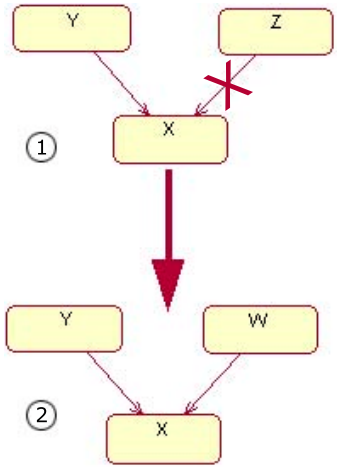

Figura 4.10: Alteração da árvore de dependência da tarefa X.

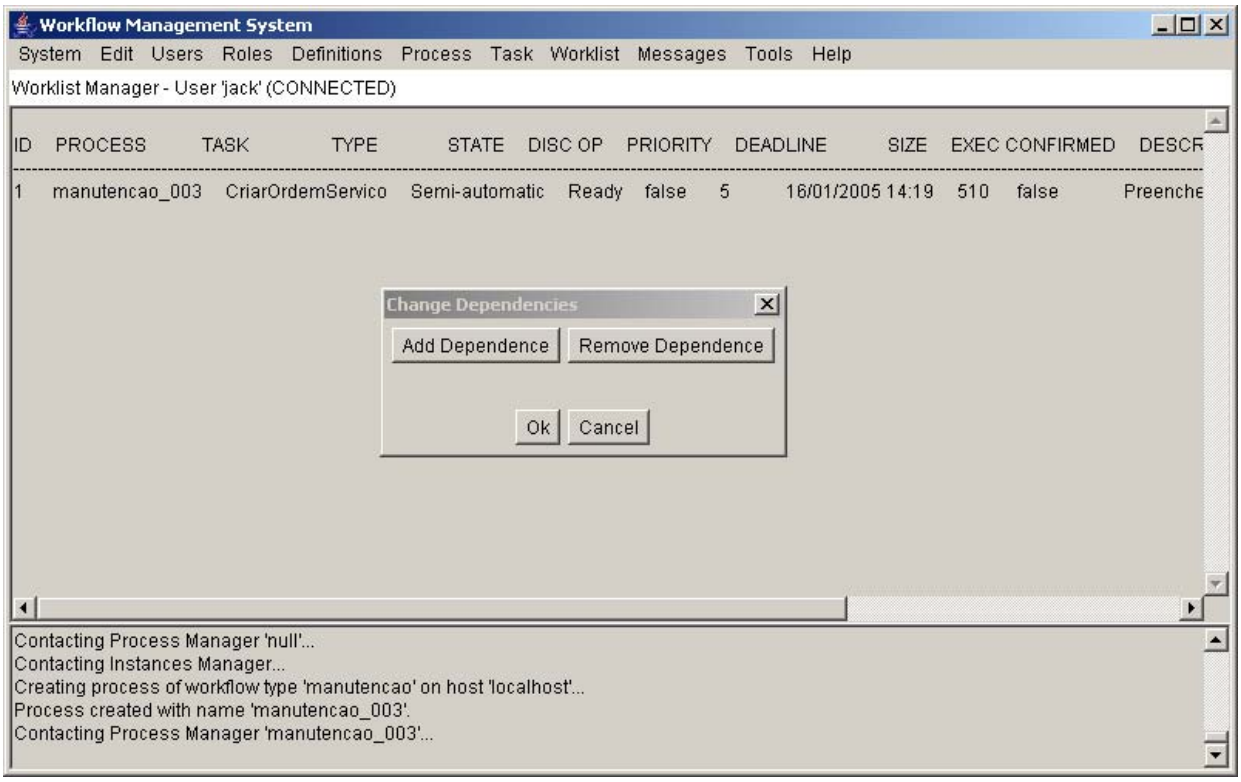

Figura 4.11: Alteração da árvore de dependência no protótipo implementado.

Um problema que pode acontecer nesta operação é alguma alteração de dependências errada ou negligente, tal alteração, pode acarretar em problemas como: *deadlock* entre tarefas do processo, particionamento do grafo que representa o processo, fazendo com que partes do processo

nunca mais sejam alcançadas. Tal problema pode ser solucionado pela inclusão de um mecanismo de verificação de consistência e particionamento. Este seria responsável por verificar o estado global do processo a cada nova alteração de dependências, para garantir que os problemas acima citados sejam minimizados. Este mecanisno não é parte inclusa deste protótipo, e poderia servir como inspiração para novos trabalhos na área.

Alterar Estado de Tarefa

O WorkToDo Flex implementa esta operação somente para aquelas tarefas que estão no estado READY, NOT_READY ou READY_SUSPENDED.

No caso de alteração de estado de uma tarefa X, podem ocorrer as seguintes situações:

- **Se o Estado de X for READY**. A tarefa X só poderá passar para os estados NOT_READY, CANCELLED ou READY_SUSPENDED.
- **Se o Estado de X for NOT_READY**. A tarefa X só poderá passar para os estados READY, CANCELLED ou READY_SUSPENDED.
- **Se o Estado de X for READY_SUSPENDED**. A tarefa X só poderá passar para os estados READY, NOT_READY ou CANCELLED.

Em todos os casos, se for escolhido o novo estado CANCELLED, a operação pode ser realizada de duas formas:

- **CASCADE**. Nesta opção o sistema altera o estado de X para CANCELLED e, recursivamente, altera os estados de toda sua árvore de dependência para CANCELLED, fazendo com que todas as tarefas que fazem parte do ramo de dependências de X sejam canceladas. A Figura 4.12 ilustra os passos deste processo;
- **NOT_CASCADE**. Nesta opção o sistema altera somente o estado de X para CANCELLED.

4.2.2 Flexibilidade por Seleção

No WorkToDo Flex, a flexibilidade por seleção é obtida através de uma tarefa especial de construção. Essa tarefa é utilizada na modelagem de subworkflows em tempo de execução, ou o que pode ser chamado de modelagem tardia (seção 2.2.1). Esta tarefa é definida no WorkToDo Flex como um modelo de tarefa igual a qualquer outro. A seguir é apresentada a definição do modelo de tarefa para a tarefa de construção:

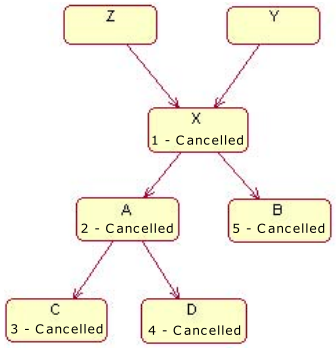

Figura 4.12: Cancelamento em Cascata.

```
TASK TaskBuilder {
        TYPE builder;
        ROLE taskbuilder;
        DESCRIPTION "TaskBuilder";
        APPLICATION;
        PRIORITY;
        DEADLINE;
        RETRIES;
}
```

Como se pode observar, foi criado um novo tipo de tarefa além dos três primitivos (automática, semi-automática e manual), que é o tipo *builder* (Construtor). Além disso, um novo papel (*taskbuilder*) também deve ser criado e um usuário relacionado ao mesmo de modo que somente usuários com este papel podem selecionar tarefas de construção. Este papel também é associado ao usuário responsável no momento da criação da instância.

Este modelo de tarefa, então, pode ser instanciado normalmente como qualquer tarefa. Ele é identificado por um nome consistindo de definição de dependências, porém sem definir contextos de entrada e saída. A seguir, é apresentado um exemplo de definição de processo com três tarefas, sendo que uma é uma tarefa de construção:

```
WORKFLOW wfexemplo {
        FILE Arquivo1 {
                NAME "arquivo1.txt";
                SIZE 10;
        }
        FILE Arquivo2 {
                NAME "arquivo2.txt";
                SIZE 20;
        }
        TASK T1: Modelo1 {
                DEPENDS;
                IN_CONTEXT arquivo1;
                OUT_CONTEXT;
        }
        TASK TB: TaskBuilder {
                DEPENDS (and (t1 -> SUCCESS));
```

```
                IN_CONTEXT;
                OUT_CONTEXT;
        }
        TASK T2: Modelo2 {
                DEPENDS (and (TB -> SUCCESS));
                IN_CONTEXT arquivo1;
                OUT_CONTEXT arquivo2;
        }
}
```

A Figura 4.13 mostra o fluxograma do processo acima especificado. A tarefa de construção T_B foi definida usando o modelo *TaskBuilder* sendo que a mesma depende do sucesso da tarefa T_1 para ser executada.

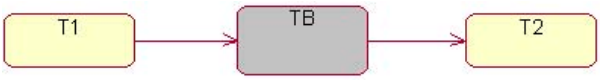

Figura 4.13: Processo que utiliza tarefas de construção.

No momento em que a tarefa de construção é selecionada pelo usuário do papel apropriado, a tarefa T_B muda para o estado READY. A seguir é exibida uma janela para a definição do subprocesso conforme a Figura 4.14. Nesta janela podem ser criadas inúmeras tarefas, suas dependências e os contextos de entrada e saída.

No início da definição do novo subworkflow a primeira tarefa é definida sem dependências, para que logo que o subworkflow entre em execução a mesma seja a primeira a se tornar READY. Portanto, a única restrição do sistema é que a primeira tarefa a ser definida para o subworkflow deve ser a tarefa de entrada na "caixa preta".

Após criada a primeira tarefa, novas tarefas podem ser criadas e as relações de dependências entre as mesmas definidas, bem como os contextos de entrada e saída de cada uma delas. A Figura 4.15 mostra o fluxograma da nova instância em execução após a construção de um possível subworkflow. A tarefa T_3 é a tarefa de entrada e as tarefas T_4 e T_5 são tarefas de saída.

O subworkflow criado funciona de maneira semelhante a um workflow normal. A única diferença, está no encerramento do subworkflow, pois neste momento a tarefa de saída da "caixa preta" (Tarefa de Construção) deve mudar para o estado READY, indicando que o subworkflow definido anteriormente terminou sua execução e também para que o fluxo de execução definido em tempo de especificação siga normalmente. A Figura 4.16 mostra a tela do protótipo após a construção do novo subworkflow.

Após esta descrição, fica claro que especificar tarefas de construção no WorkToDo Flex é simples, pois a especificação das mesmas é feita da mesma forma que a especificação de qualquer outra tarefa.

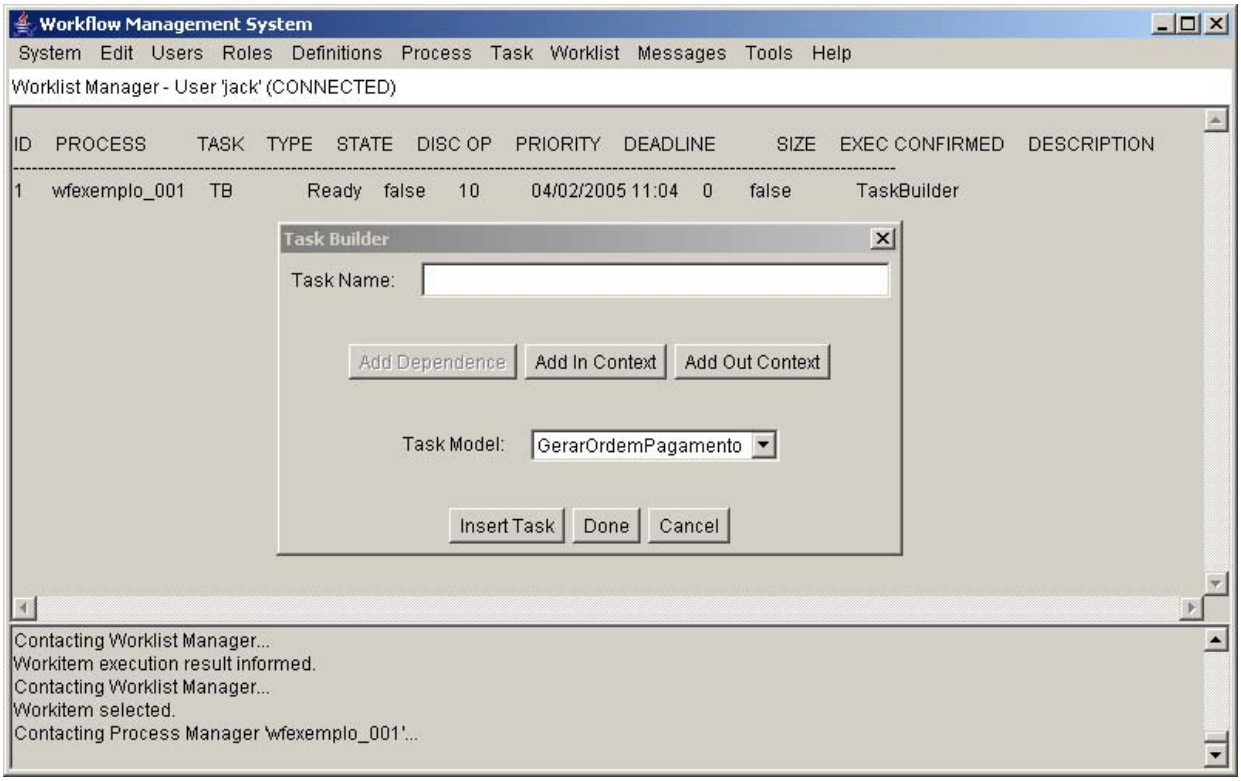

Figura 4.14: Janela do protótipo para especificação de subprocesso.

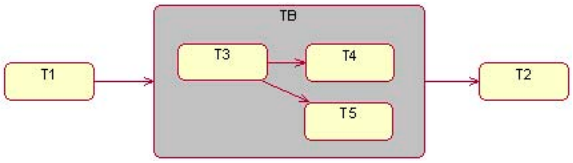

Figura 4.15: Fluxograma do workflow exemplo incluindo o novo subworkflow construído.

4.3 Arquitetura do WorkToDo Flex

A arquitetura do WorkToDo Flex (Figura 4.17) é praticamente a mesma utilizada e implementada no WorkToDo [21][22]. A principal diferença é que a comunicação entre os componentes da mesma é feita utilizando-se a invocação remota de métodos fornecida pelo RMI.

Na Figura 4.17 os componentes em destaque (hachurados) foram os que sofreram as principais alterações. A principal alteração foi a inclusão de um novo componente chamado de Gerenciador de Sub-Processos (SPM - Sub-Process Manager), o qual tem a função de gerenciar a criação e execução dos subprocessos criados pela tarefa de construção. Sua implementação é bastante semelhante ao PM.

O PM também sofreu alterações para acomodar as operações de flexibilidade e gerenciar a

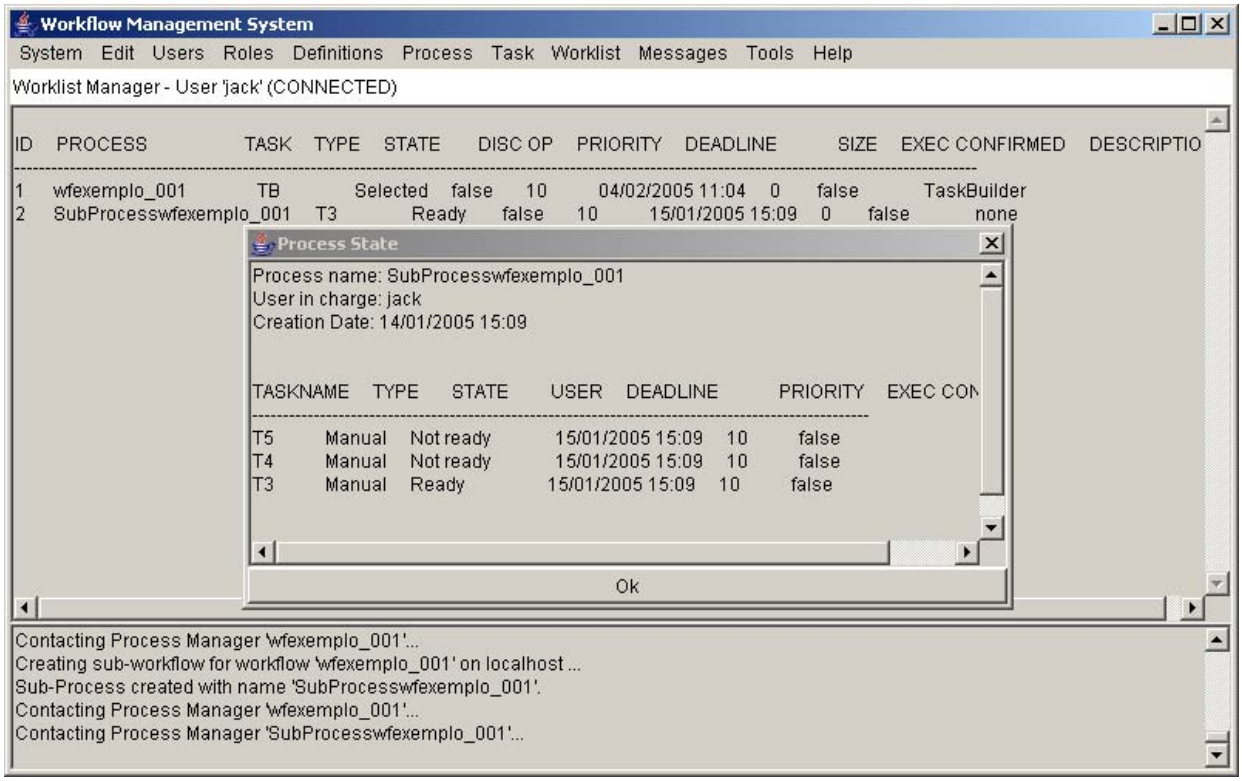

Figura 4.16: Janela do protótipo com o subworkflow construído.

criação de novos sub-processos. Já o DM teve uma simples alteração para acomodar um novo modelo de tarefa na linguagem de especificação, que é a tarefa de construção.

O WM sofreu várias alterações para incluir as novas funcionalidades de flexibilidade, como novos itens de menu, interfaces para definição de tarefas de construção, alteração de dependências, remoção de tarefas e inserção de tarefas. Além disso foram implementados novos métodos para receber e enviar as devidas notificações ao PM relativas às operações de flexibilidade.

4.4 Limitações do WorkToDo Flex

Com o objetivo de delimitar o escopo do presente trabalho, apresentaremos algumas limitações do WorkToDo Flex, quanto ao modelo proposto:

- O modelo apresentado não prevê o controle e gerenciamento de versões e variantes. Tal controle poderia ser realizado através da inserção de um novo componente na arquitetura, de forma que a cada nova alteração realizada, fosse possível armazenar uma nova versão da definição do processo, ou criar uma nova variante para esta definição.

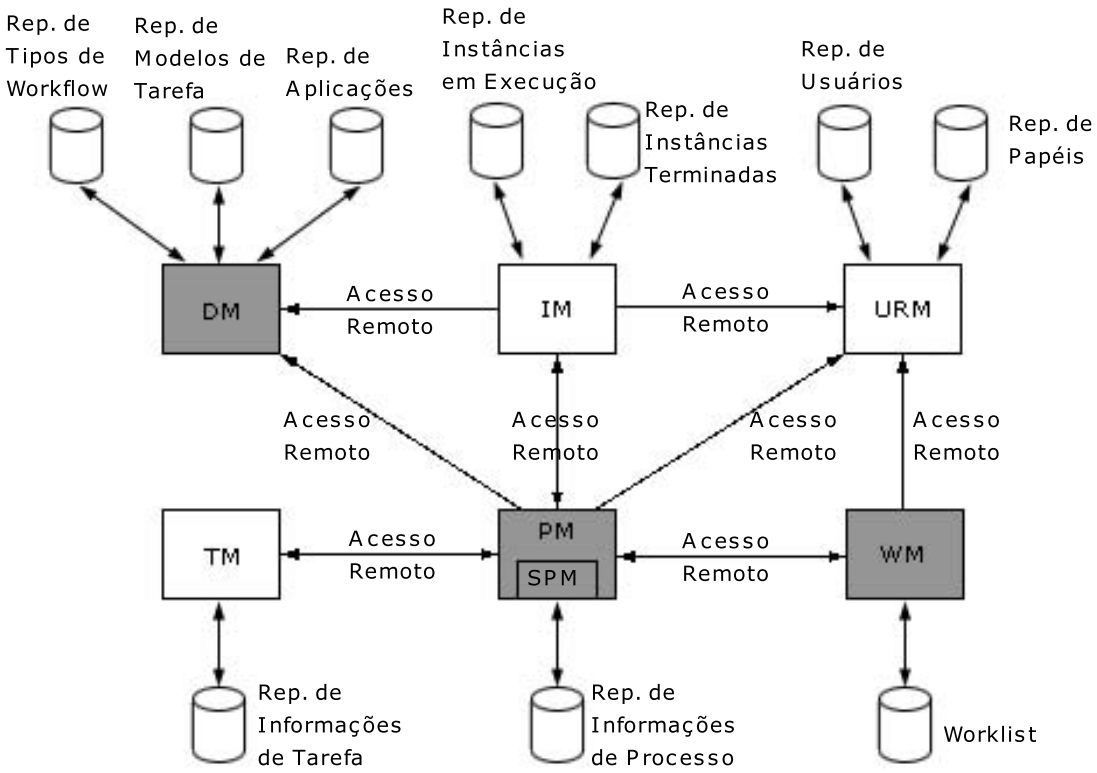

Figura 4.17: Arquitetura do WorkToDo Flex.

- O modelo não fornece a flexibilidade por adaptação de tipo, já que só trabalha com as instâncias em execução.

- O modelo não fornece mecanismos de verificação de corretude das definições de processo e suas alterações. Tal mecanismo poderia ser realizado através da inserção de um novo componente na arquitetura, de forma que a cada nova definição de processo a consistência e corretude do mesmo fosse validada. Outro aspecto está na execução das operações de adaptação, ou seja, este mecanismo seria utilizado para validar também as novas variantes e/ou versões do processo em questão.

- O modelo não fornece adaptação da definição de tarefas e aplicações, este é um mecanismo importante na mudança de um processo, porém causa complicações no controle de restrições temporais de prazos entre tarefas.

- O modelo não fornece mecanismos de recuperação de estados anteriores a uma alteração. Tal mecanismo daria origem a um novo componente da arquitetura, responsável por armazenar um histórico da execução de todos os processos. Este componente trabalharia juntamente com um possível controle de versões e variantes, já que ambos compartilha-

riam as informações de históricos dos processos.

4.5 Diagramas de Seqüência

Esta seção apresenta os diagramas de seqüência das operações de flexibilidade do WorkToDo Flex, com o objetivo de tornar mais claro o funcionamento do sistema e discutir alguns detalhes importantes.

As operações sempre se iniciam por meio de uma requisição do usuário à interface do sistema, que então encaminha a requisição ao componente adequado. Ao final, a interface retorna uma notificação ao usuário, indicando o sucesso ou falha da operação.

4.5.1 Cancelar Workflow

O procedimento para o cancelamento de uma instância de processo, mostrado na Figura 4.18, é detalhado a seguir:

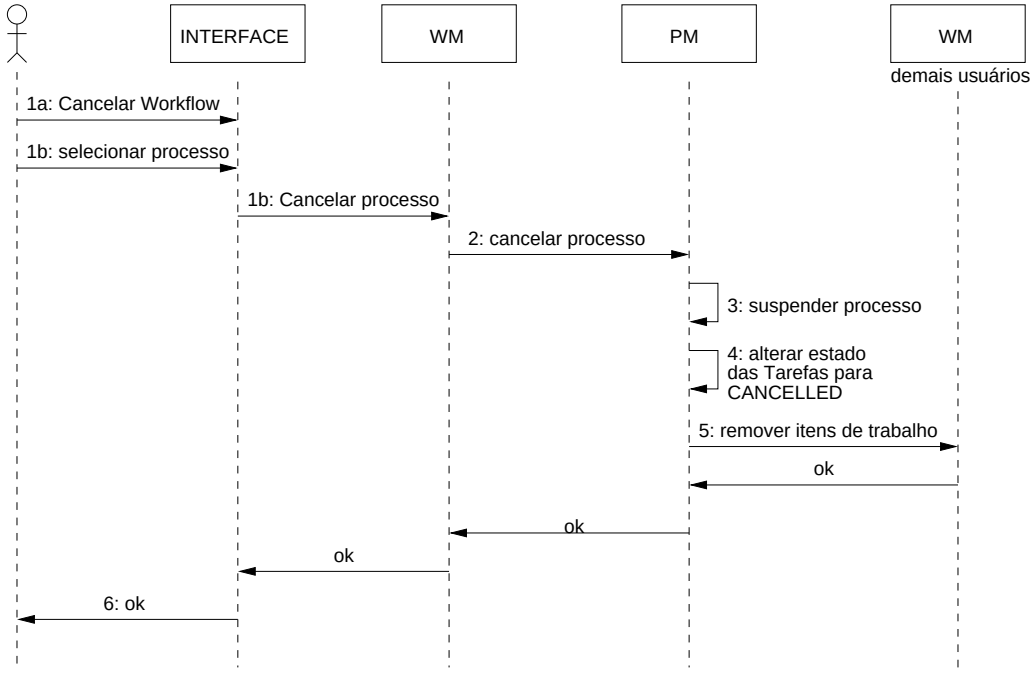

Figura 4.18: Diagrama de seqüência para o cancelamento de uma instância de processo.

1. Através da interface do sistema, o usuário requisita ao WM que uma determinada instância de processo seja cancelada;
2. O WM envia uma requisição ao PM solicitando o cancelamento da instância selecionada;

3. O PM suspende o processo;
4. O PM altera o estado de todas as tarefas (no estado READY, READY_SUSPENDED ou NOT_READY) para o estado CANCELLED;
5. O PM notifica os WMs de todos os usuários informando o cancelamento do processo;
6. A interface notifica o usuário de que o processo foi cancelado.

4.5.2 Suspender Workflow

O procedimento para a suspensão de uma instância de processo, mostrado na Figura 4.19, é detalhado a seguir:

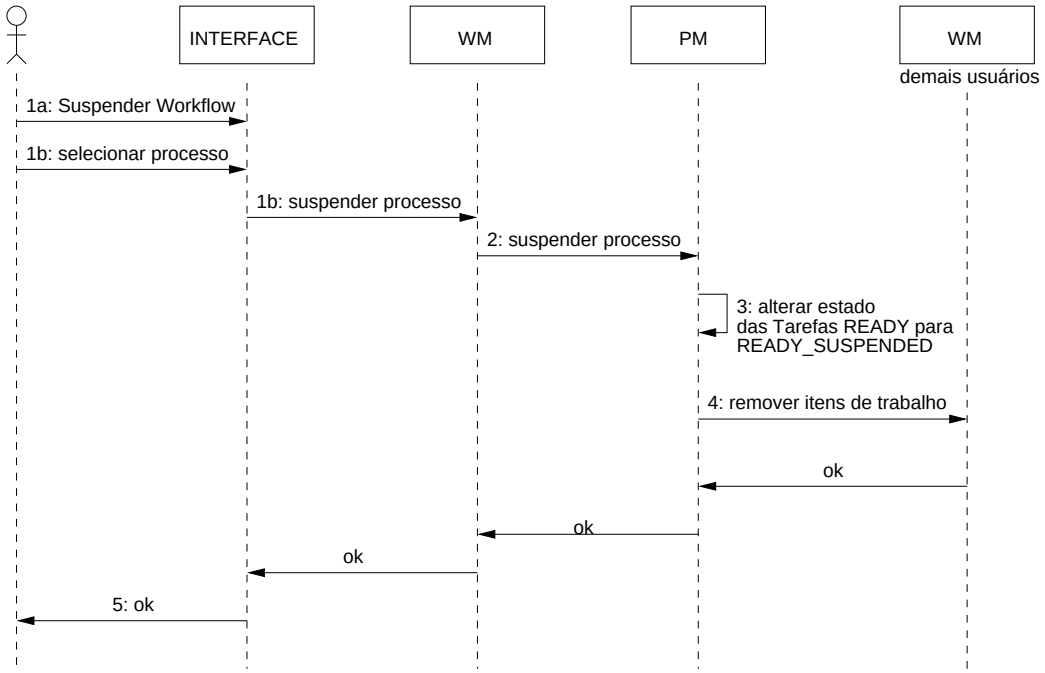

Figura 4.19: Diagrama de seqüência para o suspensão de uma instância de processo.

1. Através da interface do sistema, o usuário requisita ao WM que uma determinada instância de processo seja suspensa;
2. O WM envia uma requisição ao PM solicitando a suspensão da instância selecionada;
3. O PM altera o estado de todas as tarefas (no estado READY) para o estado READY_SUSPENDED;
4. O PM notifica os WMs de todos os usuários informando a suspensão do processo;
5. A interface notifica o usuário de que o processo foi suspenso.

4.5.3 Reiniciar Workflow

O procedimento para reiniciar uma instância de processo, mostrado na Figura 4.20, é detalhado a seguir:

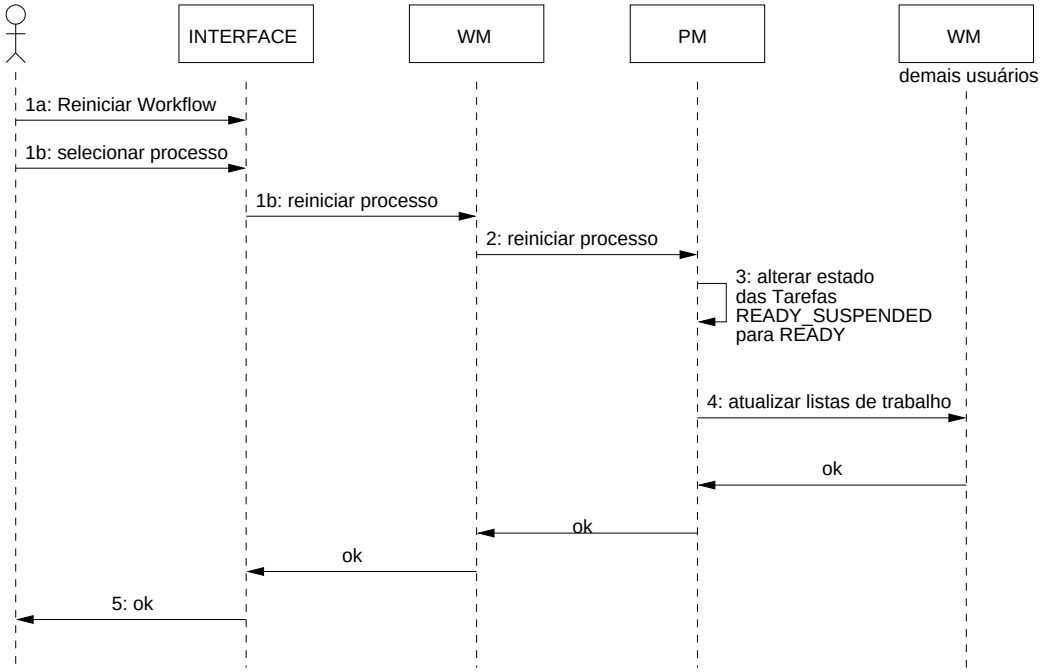

Figura 4.20: Diagrama de seqüência para reiniciar uma instância de processo.

1. Através da interface do sistema, o usuário requisita ao WM que uma determinada instância de processo seja reniniada;
2. O WM envia uma requisição ao PM solicitando o reinicio da instância selecionada;
3. O PM altera o estado de todas as tarefas (no estado READY_SUSPENDED) para o estado READY;
4. O PM notifica os WMs de todos os usuários informando o reinicio do processo;
5. A interface notifica o usuário de que o processo foi reiniciado.

4.5.4 Inserir nova tarefa

O procedimento para inserir uma nova tarefa a uma instância de processo, mostrado na Figura 4.21, é detalhado a seguir:

1. Através da interface do sistema, o usuário requisita ao WM a inserção de uma nova tarefa a um determinada instância de processo, nomeando a nova tarefa e adicionando suas dependências;

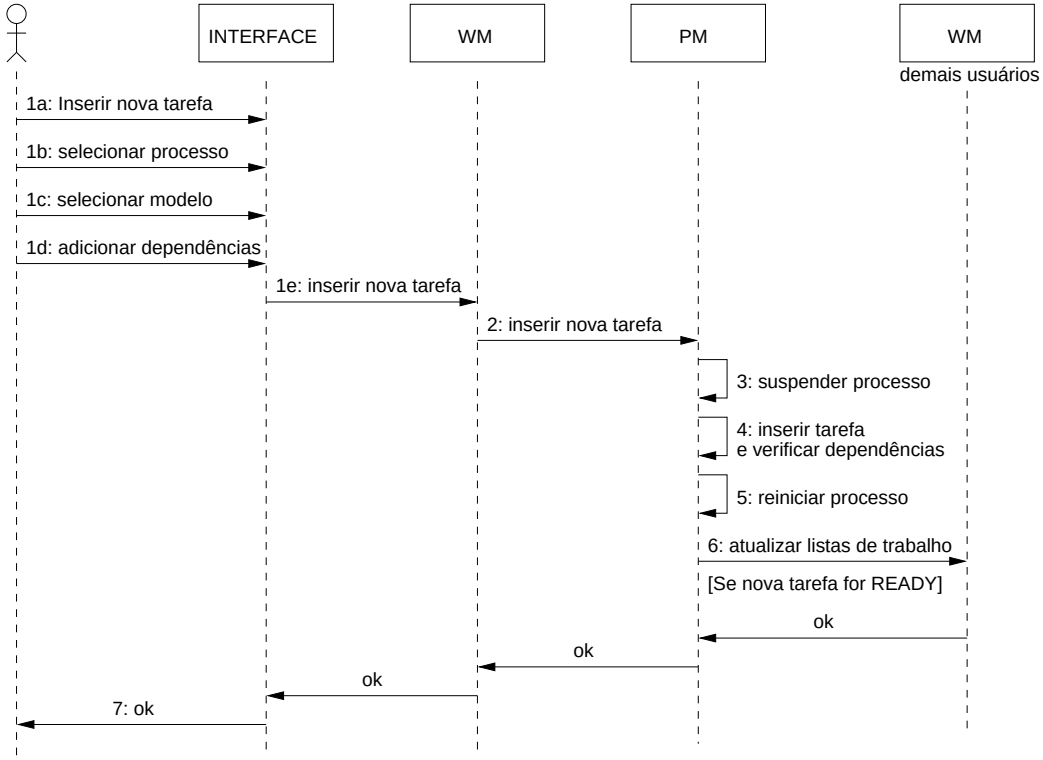

Figura 4.21: Diagrama de seqüência para inserção de nova tarefa.

2. O WM envia uma requisição ao PM solicitando a inserção da nova tarefa;
3. O PM suspende o processo;
4. O PM insere a nova tarefa e analisa sua árvore de dependências para verificar se a nova tarefa pode ficar pronta (no estado READY);
5. O PM renicia o processo;
6. O PM notifica os WMs de todos os usuários informando o novo estado do processo;
7. A interface notifica o usuário de que o processo foi realizado com sucesso.

4.5.5 Remover tarefa

O procedimento para remover uma tarefa de uma instância de processo, mostrado na Figura 4.22, é detalhado a seguir:

1. Através da interface do sistema, o usuário requisita ao WM a remoção de tarefa de uma determinada instância de processo;
2. O WM envia uma requisição ao PM solicitando a remoção da tarefa;
3. O PM suspende o processo;

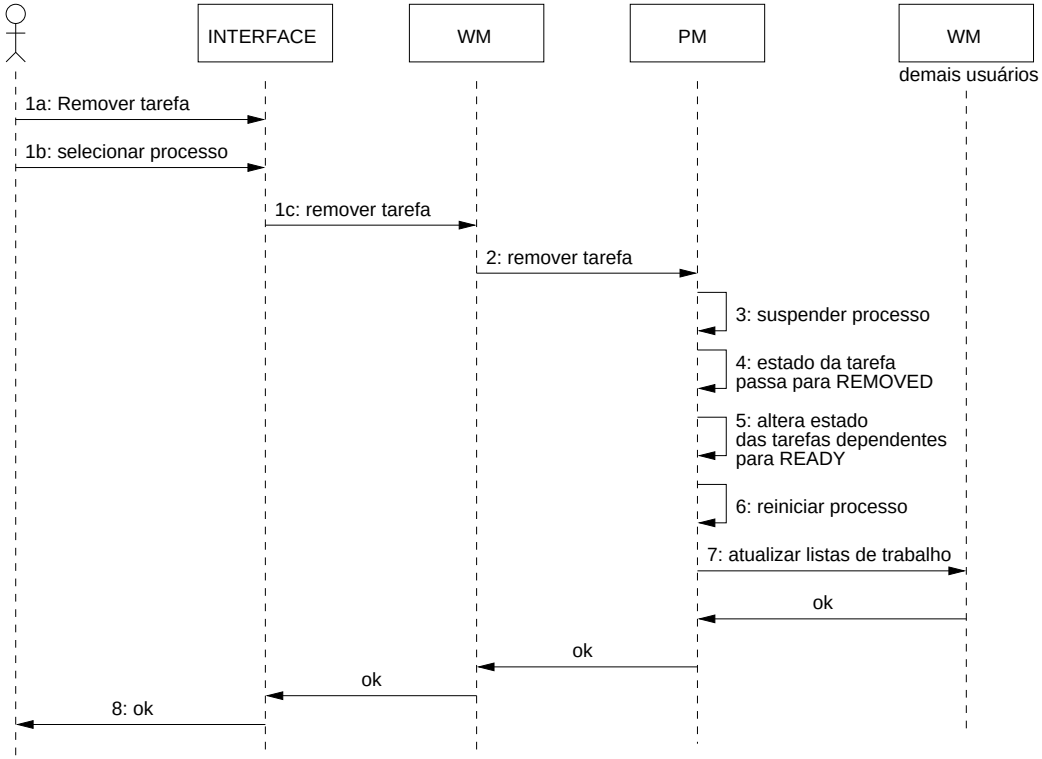

Figura 4.22: Diagrama de seqüência para remoção de tarefa.

4. O PM altera o estado da tarefa a ser removida para REMOVED;
5. O PM altera o estado de todas as tarefas dependentes para READY;
6. O PM renicia o processo;
7. O PM notifica os WMs de todos os usuários informando o novo estado do processo;
8. A interface notifica o usuário de que o processo foi realizado com sucesso.

4.5.6 Alterar estado de tarefa

O procedimento para alterar o estado de uma tarefa em uma instância de processo, mostrado na Figura 4.23, é detalhado a seguir:

1. Através da interface do sistema, o usuário requisita ao WM a alteração de estado de tarefa selecionando a instância de processo, a tarefa a ser alterada e o novo estado desejado;
2. O WM envia uma requisição ao PM solicitando a alteração do estado da tarefa;
3. O PM suspende o processo;
4. O PM altera o estado da tarefa para o solicitado;
5. O PM renicia o processo;

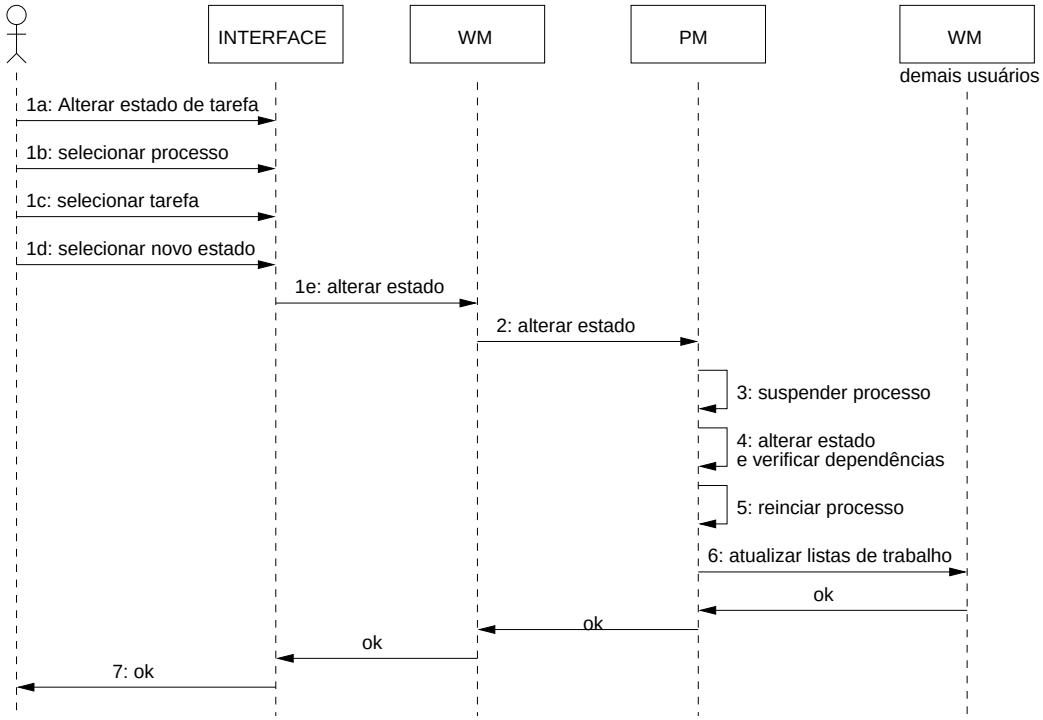

Figura 4.23: Diagrama de seqüência para alteração de estado de tarefa.

6. O PM notifica os WMs de todos os usuários informando o novo estado do processo;
7. A interface notifica o usuário de que o processo foi realizado com sucesso.

Caso o novo estado da tarefa seja CANCELLED o usuário pode escolher por executar uma alteração de estado em cascata. Procedimento pelo qual todas as tarefas dependentes terão seu estado alterado para CANCELLED. Este procedimento é realizado de maneira recursiva na árvore de dependentes da tarefa selecionada para alteração de estado.

4.5.7 Alterar dependências de tarefa

O procedimento para alterar as dependências de uma tarefa em uma instância de processo, mostrado na Figura 4.24, é detalhado a seguir:

1. Através da interface do sistema, o usuário requisita ao WM a alteração das dependências de tarefa selecionando a instância de processo, a tarefa a ser alterada e as novas dependências escolhidas;
2. O WM envia uma requisição ao PM solicitando a alteração das dependências da tarefa;
3. O PM suspende o processo;
4. O PM gera uma nova árvore de dependências;

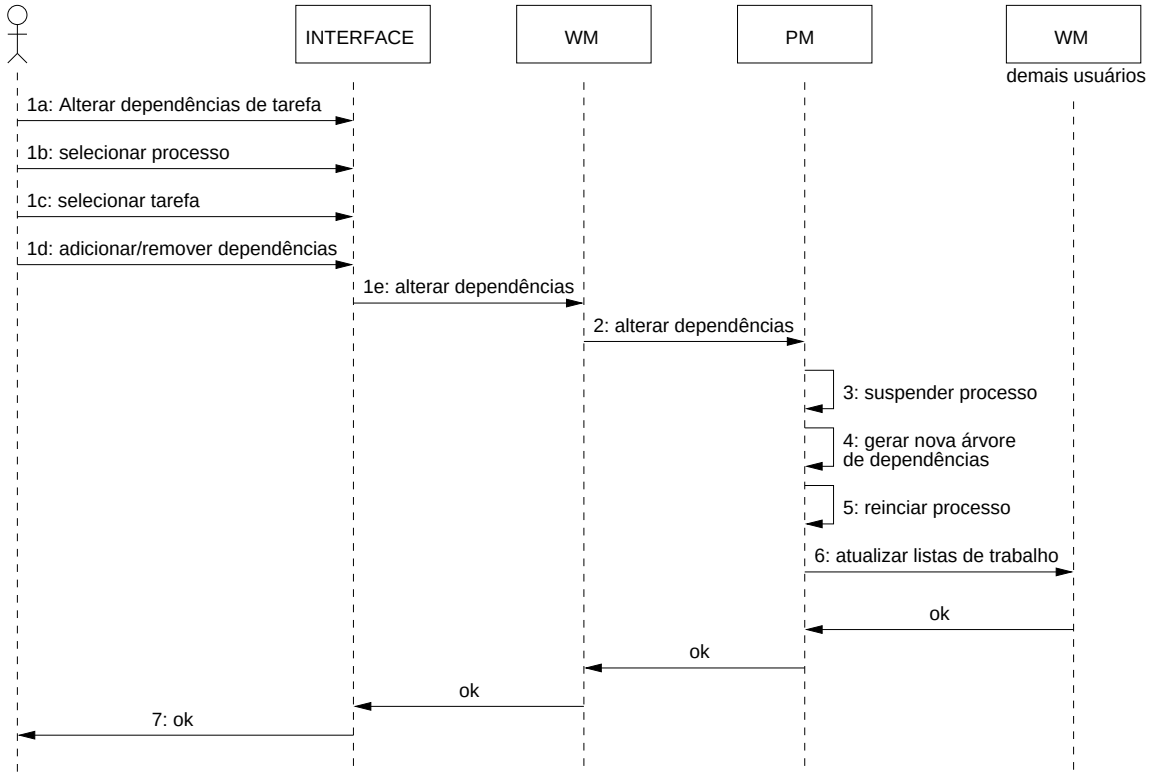

Figura 4.24: Diagrama de seqüência para alteração de dependências de tarefa.

5. O PM renicia o processo;
6. O PM notifica os WMs de todos os usuários informando o novo estado do processo;
7. A interface notifica o usuário de que o processo foi realizado com sucesso.

4.6 Diagramas de Classe

A seguir serão apresentados os diagramas de classe do sistema WorkToDo Flex. Como a implementação foi toda ela baseada em uma implementação já existente em CORBA - OrbixWeb, neste capítulo somente serão apresentados os diagramas de classe dos pacotes que sofreram as maiores alterações. Os demais pacotes não apresentados sofreram alterações em suas interfaces para implementar as chamadas remotas de métodos através do RMI e algumas adaptações mínimas para dar apoio às funcionalidades de flexibilidade implementadas.

O pacote pMgr (Figura 4.25) implementa, basicamente, quatro funcionalidades do sistema de gerenciamento de workflow. A primeira, é o analisador responsável por interpretar as especificações e estabelecer as relações de dependências entre as tarefas. Esta funcionalidade é implementada pelo pacote parser.

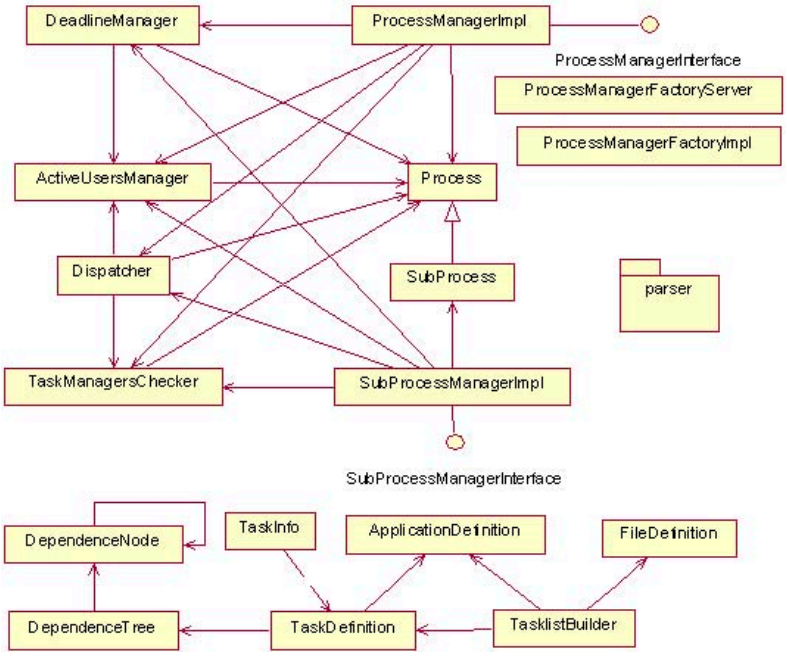

Figura 4.25: Diagrama de Classes do Pacote pMgr (Process Manager) do WorkToDo Flex.

A segunda, é responsável pela criação e monitoramento dos processos, bem como notificação de usuários, monitoramento de prazos de tarefas e encaminhamento de execução de tarefas automáticas. Esta funcionalidade é implementada pelas classes Process, ActiveUsersManager, Dispacher, TaskManagerChecker e DeadlineManager.

A terceira, é responsável pela criação da lista de trabalho e relações de dependência entre as tarefas, feita a partir da interpretação realizada pelo analisador. Esta funcionalidade é implementada pela classe TaskListBuilder. Além disso, o pacote pMgr fornece interfaces para comunicação remota com outros componentes da arquitetura.

A quarta, é responsável pela criação e monitoramento dos sub-processos criados pelas tarefas de construção. As mesmas funcionalidades de um processo normal são implementadas, com exceção do encerramento do sub-processo. Neste momento além do encerramento normal como uma instância, o estado da tarefa de construção é alterado para SUCCEEDED indicando que a construção foi executada com sucesso e o subprocesso por ela construído terminou sua execução. Esta funcionalidade é implementada pelas classes subprocess (Classe derivada de Process) e SubProcessManagerImpl.

O pacote sysInterface (Figura 4.26) implementa classes utilizadas nas aplicações dos usuários, fornecendo classes que implementam a janela principal do sistema (Classe MainWindow) e janelas secundárias usadas em operações dentro da aplicação (Classes SimpleQueryDialog, ComboBoxDialog, InformatioDialog, OptionsWindow e TextTable). É a partir da classe

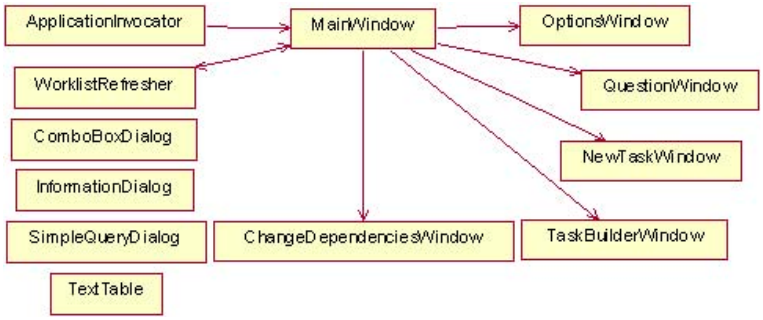

Figura 4.26: Diagrama de Classes do Pacote sysInterface (Interface Gráfica) do WorkToDo Flex.

MainWindow que são requisitadas todas as operações implementadas no sistema WordToDo.

Além das classes já mencionadas, novas classes foram implementadas para fornecer interfaces visuais para as operações de flexibilidade. São elas: QuestionWindow, NewTaskWindow, TaskBuilderWindow e ChangeDependenciesWindow.

A tabela 4.1 apresenta o número de linhas de código e classes escritas nos pacotes do WorkToDo Flex. Estes dados refletem, primeiro, a utilização de RMI que na maioria dos casos reduziu o número de classes utilizadas e em alguns casos o número de linhas escritas. Segundo, devido a implementação de novas funcionalidades o total de linhas escrito foi maior. Este dado pode ser observado com maior detalhe nos pacotes onde o maior número de alterações foram realizadas (PM, WM, TM e Interface).

	DM		IM	
	WorkToDo	WorkToDo Flex	WorkToDo	WorkToDo Flex
Linhas de Código	495	499	472	397
Número de classes	6	6	3	3

	URM		PM	
	WorkToDo	WorkToDo Flex	WorkToDo	WorkToDo Flex
Linhas de Código	836	813	2931	3564
Número de classes	7	7	16	18

	TM		WM	
	WorkToDo	WorkToDo Flex	WorkToDo	WorkToDo Flex
Linhas de Código	305	351	1747	1834
Número de classes	5	4	9	7

	INTERFACE		UTIL	
	WorkToDo	WorkToDo Flex	WorkToDo	WorkToDo Flex
Linhas de Código	1832	3352	1684	1936
Número de classes	7	13	12	14

	TOTAL	
	WorkToDo	WorkToDo Flex
Linhas de Código	10302	12746
Número de classes	65	72

Tabela 4.1: Linhas de código do protótipo.

Capítulo 5

Conclusões

Nesta dissertação, foi apresentado o WorkToDo Flex, um SGWf para workflows flexíveis. O sistema fornece dois tipos de flexibilidade: por adaptação e por seleção. A flexibilidade por adaptação é fornecida por um conjunto de operações realizadas em instâncias de processo em execução. A flexibilidade por seleção é obtida pela especificação e execução de tarefas de construção.

As operações de flexibilidade por adaptação são as seguintes: Cancelar Workflow, Suspender Workflow, Reiniciar Workflow, Inserir nova Tarefa, Remover Tarefa, Alterar Dependências de Tarefa e Alterar Estado de Tarefa. As três primeiras operações causam impacto no estado de todas as tarefas da instância. As operações de inserir, remover, alterar dependências e alterar estado de tarefa manipulam as tarefas do processo de maneira isolada, ou seja, cada operação só manipula uma tarefa por vez. Mas a remoção de uma tarefa causa uma mudança de estado para READY em todas as tarefas que dependem da tarefa removida.

A especificação e execução de tarefas de construção são fornecidas para implementar a flexibilidade por seleção. Tal funcionalidade usa o conceito de "caixa preta", onde são definidos pontos de construção dentro da definição do processo. Nestes pontos são definidas estas tarefas de construção, que tem por função construir subworkflows em tempo de execução. Esta funcionalidade permite que a cada nova instância de processo criada, novos fluxos de execução possam ser criados e executados. Além disso, permite que o usuário responsável possa definir novos subworkflows sem precisar alterar a definição do processo global.

Apresentamos também um protótipo para avaliar as soluções propostas. O sistema leva em conta a distribuição dos componentes e usuários, já que workflows envolvem vários usuários e aplicativos localizados em diferentes pontos de um sistema distribuído. Para antender tal necessidade, o protótipo foi implementado em Java sobre a plataforma distribuída RMI. Além disso, essa plataforma permite a execução do SGWf em diferentes sistemas operacionais e plataformas de hardware aumentando sua flexibilidade de uso.

5.1 Contribuições

Este trabalho teve como principais contribuições:

- A apresentação de um modelo e arquitetura de SGWf para workflows flexíveis.
- Um conjunto de operações simples para prover flexibilidade por adaptação.
- A incorporação de tarefas de construção na definição do processo para prover flexibilidade por seleção.
- A implementação de um protótipo como forma de avaliar o modelo proposto.

5.2 Trabalhos Futuros

Podemos identificar alguns trabalhos futuros, descritos a seguir:

- Incorporar ao modelo e arquitetura componentes responsáveis para realizar o controle e gerenciamento de versões e variantes. A implementação deste componente seria de grande utilidade na evolução do modelo, já que o controle e gerenciamento de versões em workflows flexíveis é um aspecto que proporciona maior flexibilidade ao SGWf.
- Incorporar ao modelo a flexibilidade por adaptação de tipo, dando ao usuário responsável a possibilidade de escolher quais instâncias serão afetadas por uma mudança.
- O armazenamento dos repositórios em um banco de dados ao invés de arquivos, a fim de aumentar a confiabilidade e segurança das informações.
- O desenvolvimento de linguagens gráficas para especificação e visualização das definições de tipos de processos, tarefas, aplicações e tarefas de construção.
- O desenvolvimento de ferramentas de administração que facilitem o gerenciamento do sistema.
- A implementação de políticas de prioridade de tarefas no Despachante.
- A implementação de alguns aspectos que não foram implementados no protótipo, tais como a utilização de dados do tipo Consulta SQL em banco de dados, entre outros.
- Ampliação do módulo de gerenciamento de usuários e papéis que incorpore o controle de níveis de acesso mais detalhados ao sistema.

- Incorporar ao modelo componentes responsáveis pelo controle e gerenciamento de Agentes Móveis capazes de executar tarefas automáticas que necessitam de dados que não estão disponíveis para o SGWf.
- Incluir mecanismos para verififar a corretude das definições de processo e suas alterações. Pois nestas definições podem ser criados *deadlocks*, particionamentos do grafo na alteração de dependências e instâncias que não atingem o estado final.

Referências Bibliográficas

[1] Aalst, W. M. P. van der e S. Jablonski: *Dealing With Workflow Change: Identification of Issues and Solutions*. International Journal of Computer Systems Science and Engineering, 15(5), Setembro 2000, ISSN 0267-6192.

[2] Alonso, G., D. Agrawal, A. El Abbadi, R. Günthör, M. Kamath, e C. Mohan: *Advanced Transaction Models in Workflow Context*. In *Proceedings of the 12th International Conference on Data Engineering*. New Orleans, USA, 1996.

[3] Alonso, G., D. Agrawal, A. El Abbadi, e C. Mohan: *Functionality and Limitations of Current Workflow Management Systems*. Relatório Técnico, IBM Almaden Research Center, 1997.

[4] Barbará, D., S. Mehrotra, e M. Rusinkiewicz: *INCAs: A Computation Model for Dynamic Workflows in Autonomous Distributed Environments*. Relatório Técnico, Department of Computer Science, University of Houston, 1994.

[5] Barbará, D., S. Mehrotra, e M. Rusinkiewicz: *INCAs: Managing Dynamic Workflows in Distributed Environments*. Journal of Database Management, 7(1), 1996.

[6] Bogia, Douglas P. e Simon M. Kaplan: *Flexibility and Control for Dynamic Workflows in the WORLDS Environment*. In *Proceedings of conference on Organizational computing systems*, páginas 148–159. ACM Press, 1995, ISBN 0-89791-706-5.

[7] CORBA: *Common Object Request Broker Architecture - http://www.corba.org*, Agosto 2004.

[8] Eder, J., H. Groiss, e W. Liebhart: *Workflow Management and Databases*. In *Proceedings of the 2nd Forum International d'Informatique Appliquee*. Tunis, 1996.

[9] Ellis, Clarence A. e Karim Keddara: *A Workflow Change Is a Workflow*. In *Business Process Management, Models, Techniques, and Empirical Studies*, páginas 201–217. Springer-Verlag, 2000, ISBN 3-540-67454-3.

[10] Georgakopoulos, D., M. Hornick, e A. Sheth: *An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure*. Distributed and Parallel Databases, 3:119–153, 1995.

[11] Heinl, Petra, Stefan Horn, Stefan Jablonski, Jens Neeb, Katrin Stein, e Michael Teschke: *A Comprehensive Approach to Flexibility in Workflow Management Systems*. In *Proceedings of the International Joint Conference on Work Activities Coordination and Collaboration*, páginas 79–88. ACM Press, 1999, ISBN 1-58113-070-8.

[12] IONA: *IONA Technologies - http://www.iona.com*, Agosto 2004.

[13] J2SE, Java: *Java 2 Standard Edition - http://java.sun.com/reference/api/index.html*, Agosto 2004.

[14] Jablonski, S.: *Functional and Behavioral Aspects of Process Modelling in Workflow Systems*. In G. Chroust, A. Benczur (editor): *CON 94 Workflow Management: Challenges, Paradigms and Products*, páginas 113–133. R. Oldenburg, 1994.

[15] Jablonski, S. e C. Büssler: *Workflow Management: Modelling Concepts, Architecture and Implementation*. International Tompson Computer Press, 1996.

[16] Liu, Chengfei, Maria E. Orlowska, e Hui Li: *Automating Handover in Dynamic Workflow Environments*. In *Proceedings of the 10th International Conference on Advanced Information Systems Engineering*, páginas 159–171. Springer-Verlag, 1998, ISBN 3-540-64556-X.

[17] OrbixWeb: *IONA Technologies Java ORB Implementation - http://www.iona.com/products/orbix.htm*, Agosto 2004.

[18] Reichert, Manfred e Peter Dadam: *A Framework for Dynamic Changes in Workflow Management Systems*. In *DEXA Workshop*, páginas 42–48, 1997.

[19] Reichert, Manfred e Peter Dadam: *ADEPT flex - Supporting Dynamic Changes of Workflows Without Losing Control*. Journal of Intelligent Information Systems, 10(2):93–129, 1998.

[20] Reinehr, Leonardo Hartleben: *WorkToDo - Um Sistema de Gerenciamento de Workflows para Ambientes de Comunicação Sem Fio*. Dissertação (mestrado), Universidade Estadual de Campinas, Instituto de Computação, 2002.

[21] Reinehr, Leonardo Hartleben e Maria Beatriz Felgar de Toledo: *A CORBA-based Workflow Management System for Wireless Communication Environments*. In *Proceedings of*

Confedered International Conferences: CoopIS, DOA and ODBASE, páginas 827–844, 2002.

[22] Reinehr, Leonardo Hartleben e Maria Beatriz Felgar de Toledo: *WorkToDo: Um Sistema de Gerenciamento de Workflows para Ambientes de Comunicação sem Fio*. In *Proceedings of 20th Simpósio Brasileiro de Redes de Computadores*, páginas 619–634, 2002.

[23] Sadiq, Shazia W., Wasim Sadiq, e Maria E. Orlowska: *Pockets of Flexibility in Workflow Specification*. In *Proceedings of the 20th International Conference on Conceptual Modeling*, páginas 513–526. Springer-Verlag, 2001, ISBN 3-540-42866-6.

[24] UML: *Unified Modeling Language - http://www.uml.org*, Agosto 2004.

[25] Veijalainen, J., A. Lehtola, e O. Pihlajamaa: *Research Issues in Workflow Systems*. Relatório Técnico, VTT Information Technology, Finland, 1995.

[26] WfMC: *The Workflow Reference Model*. Relatório Técnico TC00-1003, Workflow Management Coalition Group, 1995.

[27] WfMC: *Workflow Management Coalition Group - http://www.wfmc.org*, Agosto 2004.

Apêndice A

Gramática da Linguagem de Definição de Processo

A.1 Notação BNF

Esta Seção apresenta a notação BNF (*Backus Normal Form*), utilizada para descrever a gramática da Linguagem de Definição de Processo.

Notação	Significado
< . . . >	Símbolos não-terminais. Os caracteres que não são escritos entre < e > são símbolos terminais (tokens)
. . . ::= . . .	Regra de produção. O lado esquerdo contém um símbolo não terminal e o lado direito sua expansão, que pode conter símbolos terminais e não terminais
. . . | . . .	Alternativa
[. . .]	Opcional. O(s) símbolo(s) entre colchetes podem ou não aparecer na produção
x*	Repetição. O símbolo x (terminal ou não terminal) pode ser repetido 0 ou mais vezes

Tabela A.1: Descrição da notação BNF.

A.2 Tipo de Processo

```
<WORKFLOW>     ::= WORKFLOW <IDENT> { <DEFINITION>* }
<DEFINITION>   ::= FILE <IDENT> { <FILE_DEF> } |
                   QUERY <IDENT> { <QUERY_DEF> } |
                   STRING <IDENT> { <STRING_DEF> } |
                   NUMBER <IDENT> { <NUMBER_DEF> } |
                   TASK <IDENT> :<IDENT> { <TASK_DEF> } |
                   SUB-WORKFLOW <IDENT> :<IDENT> { <SUB_DEF> }
<FILE_DEF>     ::= NAME <STRING> ;
<QUERY_DEF>    ::= DATABASE <STRING> ; EXPRESSION <STRING> ;
<STRING_DEF>   ::= VALUE <STRING>;
<NUMBER_DEF>   ::= VALUE <INTEGER>;
<TASK_DEF>     ::= [<DEPENDS> ;] [<IN_CONTEXT> ;] [<OUT_CONTEXT> ;]
                   [<ROLE> ;] [<DESCRIPTION> ;]
<SUB_DEF>      ::= [<DEPENDS> ;]
<DEPENDS>      ::= DEPENDS DEPENDENCE_RULE
<IN_CONTEXT>   ::= IN_CONTEXT <IDENT> <IDENT_LIST>*
<OUT_CONTEXT>  ::= OUT_CONTEXT <IDENT> <IDENT_LIST>*
<ROLE>         ::= ROLE <IDENT>
<DESCRIPTION>  ::= DESCRIPTION <STRING>
<IDENT_LIST>   ::= , <IDENT>
<IDENT>        ::= ID
<STRING>       ::= STR
<INTEGER>      ::= INT
```

A.3 Modelo de Tarefa

```
<TASK>            ::= TASK <IDENT> { <DEFINITION> }
<DEFINITION>      ::= <TYPE> ; [<APPLICATION> ;] <PRIORITY> ;
                      [<DEADLINE> ;] <DISCONNECTED_OP> ; [<RETRIES> ;]
<TYPE>            ::= TYPE [AUTOMATIC | SEMI_AUTOMATIC | MANUAL | BUILDER]
<APPLICATION>     ::= APPLICATION <IDENT>
<PRIORITY>        ::= PRIORITY <INTEGER>
<DEADLINE>        ::= DEADLINE <INTEGER> [HOURS | DAYS]
<DISCONNECTED_OP> ::= DISCONNECTED_OPERATION <BOOLEAN>
<RETRIES>         ::= RETRIES <INTEGER>
<IDENT>           ::= ID
<INTEGER>         ::= INT
<BOOLEAN>         ::= TRUE | FALSE
```

A.4 Aplicação

```
<APPLICATION> ::= APPLICATION <IDENT> { <DEFINITION> }
<DEFINITION>  ::= <FILENAME> ; <SIZE> ; <OS> ; <CPU> ; [<INSTALLER> ;]
                  <HOSTS> ;
<FILENAME>    ::= FILENAME <STRING>
<SIZE>        ::= SIZE <INTEGER>
<OS>          ::= OS <STRING>
<CPU>         ::= CPU <STRING>
<INSTALLER>   ::= INSTALLER <STRING>
<HOSTS>       ::= HOSTS <STRING> <STRING_LIST>*
<STRING_LIST> ::= , <STRING>
<IDENT>       ::= ID
<STRING>      ::= STR
<INTEGER>     ::= INT
```

A.5 Regra de Dependência

```
<EXPRESSION>   ::= <OPERATOR> ( <EXPRESSION_1> <EXPRESSION_2>* )
<EXPRESSION_1> ::= <ITEM> | <EXPRESSION>
<EXPRESSION_2> ::= , <EXPRESSION_1>
<ITEM>         ::= <IDENT> → <STATE>
<OPERATOR>     ::= AND | OR
<STATE>        ::= READY | RUNNING | FAIL | SUCCESS
<IDENT>        ::= ID
```